Final Technical Report to the

Office of Naval Research

by

R. V. Helgason, J. L. Kennington, and K. H. Lewis
Department of Computer Science and Engineering
SOUTHERN METHODIST UNIVERSITY
School of Engineering and Applied Science
Dallas, Texas 75275-0122

for

Grid Free Algorithms for Strike Planning for Cruise Missiles

ONR Contract Number N00014-96-1-0315

SMU Contract Number 5-25182

27 February 1998

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	unrestricted
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Southern Methodist University / 6b. OFFICE SYMBOL (If applicable): CSE	7a. NAME OF MONITORING ORGANIZATION: Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code): 6425 Airline Drive, Dallas TX 75205	7b. ADDRESS (City, State, and ZIP Code): 800 North Quincy Street, Arlington VA 22217-5660
8a. NAME OF FUNDING / SPONSORING ORGANIZATION: ONR / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): 800 North Quincy Street, Arlington VA 22217-5660	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification)	
12. PERSONAL AUTHOR(S): R. Helgason, J. Kennington, and K. Lewis	
13a. TYPE OF REPORT: technical / 13b. TIME COVERED: FROM 2/1/97 TO 1/31/98	14. DATE OF REPORT (Year, Month, Day): 98 - 2 - 27 / 15. PAGE COUNT
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES: FIELD / GROUP / SUB-GROUP	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): cruise missle strike planning

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document contains two technical reports related to the development of a strike plan for a multiple cruise missle launch. The first report presents an extension of our previous algorithm for finding a short path with few segments from a designated origin to a designated destination that either avoids all threat regions or satisfies some probability constraint for a successful traversal based on proximity to the threats. In our previous work the threat regions were restricted to be circles, but based on this new work they may now also be modelled as convex polygons. The second report presents an algorithm for automatic generation of a strike plan for multiple missions which assures some reasonable path separation for missles having a common origin and destination. This new algorithm utilizes the enhanced single mission algorithm presented in the first report.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Jeffery L. Kennington	22b. TELEPHONE (Include Area Code): (214) 768-3088 / 22c. OFFICE SYMBOL: CSE

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Table of Contents

I. Statement of Work

One of the Navy's most important weapon systems is the cruise missle. Generally a missile is called a cruise missile if its speed is subsonic, if it uses a built-in global positioning navigation system (GPS/INS), and if its range is at least several hundred miles. During the early 1990s, cruise missiles launched from ships and submarines were used with great success in the Persian Gulf. Targets located hundreds of miles inland were successfully attacked by these Navy weapons stationed in relatively safe positions offshore. In addition, cruise missiles have been used to disable SAM sites, making manned aircraft strikes safer.

A mission for a cruise missile is defined as a path from the launch area to the target. The job of the mission planner is to determine the path that the missile will follow and program the missile by specifying a sequence of waypoints and including any TERCOM maps that are available. The missile uses its on-board computational facilities and its GPS/INS system to guide it to its target.

Missiles are never launched as singles and are launched in groups to form a strike. A strike plan involves developing a plan for many missiles all of which may be launched during a short period of time. Frequently two or more missiles will have the same target or different locations within the same target. For example, three missiles may be needed to hit different areas of a plant to incapacitate the plant.

Cruise missles are designed to be low flying (terrain following) which makes them extremely difficult to destroy enroute to the target. The mission planners attempt to either avoid SAM sites or fly so low that a SAM radar system can not see the incoming missile. However, cruise missiles are vulnerable to anti-aircraft artillery (triple A) if a hostile ground crew is given sufficient warning. If the triple A ground crew has its guns aimed at the correct spot, a missile can be destroyed by a single round striking a key component.

The best strategy to defeat triple A is to vary the missions of the cruise missiles. If the first missile of a strike passes directly over a triple A site, then the second should fly a few hundred yards to either the left or right.

During 1996 our research team develped algorithms designed to aid a mission planner in the mission development process. One of these algorithms was a auto-router based on geometric heuristics which performed well on randomly-generated test cases. In this research we present (1) an enhancement of the auto-router algorithm which allows greater flexibility in the modelling of the threat regions associated with SAM sites, and (2) a new algorithm designed to aid a strike planner in the development of multiple mission plans which incorporate variation in related missions.

II. Path Generation with Convex Polygonal Threats

Using current technology, missions are developed using a two-step process. In the first step, a mission planner uses a two-dimensional map to manually select a path from a launch site to the target. In the second step a software system includes the vertical dimension and develops an estimate of the probability of a successful mission. This process may be repeated several times until the mission planner is satisfied with the mission plan.

In 1996 we developed an auto-router algorithm to aid in mission planning which represented threat regions as circles. Within this framework, the problem of developing a single mission for a single cruise missile can be viewed as a computational geometry problem in which we seek a path composed of line segments from a launch site to a target site which skirts selected threat regions. The basic idea on which the algorithm is based finds a circumscribed triangle which encloses a given circle (threat) and has one vertex at a current missle position (waypoint). A missile can avoid a threat by moving along the sides of such a triangle. In empirical tests on twenty randomly generated test problems, the circumscribed triangle auto-router was found to work very well. This basic algorithm was enhanced to allow some penetration of the threat regions by the missle path while satisfing a probability constraint for a successful traversal based on proximity to the threats.

According to analysts at the Workshop for 6.1 Research on Optimization held in Dahlgren, Virginia on 1-2 October 1996, threats can come in all shapes and sizes. When routing over water, missions usually avoid islands (which can take any shape) and other ships which look more like ellipses than circles. To accomodate these more general shapes for threat regions we have enhanced the algorithm by allowing the mission planner to model any shape defined as the convex hull of given points.

A technical report giving details of the algorithm resulting from our development work appears in Appendix A. This report includes a qualitative evaluation based on visual displays of solution paths for randomly generated test problems.

III. Multiple Path Generation with Separation

The auto-routers we have developed were designed only to aid in determining a single mission for a single missile. A strike plan involves many missiles and many targets (25 missiles is not unusual). Also the probability of a safe traversal through an area is affected by multiple missiles being routed through an area. We have assumed that vulnerability to triple A can be effectively reduced by introducing a reasonable amount of separation in the flight paths of missiles at the points of nearest approach to threat regions.

To aid a strike planner we have developed an algorithm which allows the specification of several missle location and target pairs together with the specification of the number of multiple missiles to be launched for each pair. For each pair, a mission plan is developed for the first missle by applying the auto-router. Then each threat region approached by the first missle mission plan is slightly enlarged. Then a mission plan is developed for the next missle by applying the auto-router to the modified configuration. Again, any threat region approach by the next missle mission plan is slightly enlarged. Continuing in this manner assures that multiple missles will have some separation when approaching threat regions along similar paths. After all mission plans have been generated for all missles associated with a particular missle location and target pair, all threat regions are reset to their normal configuration.

A technical report giving details of the strike planning algorithm resulting from our development work appears in Appendix B. This report includes visual displays of solution paths for twenty randomly generated test problems. Each problem has five missle location and target pairs and has five missles to be launched per pair.

Appendix A

Mission Planning Path Generation

Technical Report 97-CSE-23

Cruise Missile Mission Planning: A Geometric Algorithm for Automatic Path Generation

by

R. V. Helgason
(helgason@seas.smu.edu)

J. L. Kennington
(jlk@seas.smu.edu)

K. R. Lewis
(klewis@seas.smu.edu)

Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122

November 1997

Abstract

This manuscript presents mathematical models and solution algorithms for the problem of finding a path composed of line segments from a given missile location to a given target location in the presence of threats. The threats are modelled as either circles or convex polygons. The paths must either avoid the threats or satisfy some probability constraint of a successful traversal based on the proximity of the path to the threats. The algorithms are all based on geometric concepts and their output is evaluated qualitatively based on visual displays of the problem and solution path.

Acknowledgement

This research was supported in part by the Office of Naval Research under contract number N00014-96-1-0315.

1 INTRODUCTION

During recent conflicts in the Persian Gulf, the U.S. strategy was to launch a cruise missile strike followed by manned aircraft strikes. The cruise missile strike would involve approximately 25 missiles which were launched from Navy ships and submarines stationed in relatively safe offshore positions. Communication systems and SAM sites were frequently targeted, which made any following manned strikes substantially safer.

A cruise missile strike consisting of 25 missiles requires developing a mission for each missile. A mission is defined as a path composed of line segments from a launch site to a target site. Using current technology, missions are constructed using a two-step process. In the first step, a mission planner uses a two-dimensional map to manually select a path (mission) from the launch site to the target. A software system takes this path, adds the vertical dimension, and estimates the probability of success. The probability of success is a function of the path chosen and is based on the length of the mission and the proximity of the path to known threats.

These weapons are designed to be low flying (terrain-following) which makes them extremely difficult to destroy en route to the target. The mis-

planners attempt to either avoid SAM sites or to fly so low that a SAM radar system cannot detect the incoming missile. However, cruise missiles are vulnerable to anti-aircraft artillery (AAA) if the ground crew is given sufficient warning. If the AAA ground crew has their guns aimed at the correct spot, the missile can be destroyed if a round strikes a key component. The best strategy to defeat AAA is to vary the missions of the cruise missiles. If the first missile of the strike passes directly over an AAA site, then the second should fly a few hundred yards to either the left or right.

After a mission plan has been selected for a missile, the mission planner programs the missile with this path including any TERCOM maps that will be used. A mission does not have to include TERCOM maps, but they can be used as part of the navigation system. Some missions only involve GPS waypoints.

Obviously, a graphical system which allows the mission planner to create mission plans with a point-and-click system will be an improvement over a completely manual system. However, requiring the mission planner to repeat this task 25 times to create a strike plan will be tiresome. Navy analysts indicate that a better system would provide the mission planner with a first-cut strike plan and allow the planner to modify some of the missions using a

point-and-click mechanism. To automatically create a strike plan involving 25 missiles requires 25 mission plans. Preliminary to creating such a strike plan, we must be able to generate a single cruise missile mission. The objective of this investigation is to develop algorithms to automatically generate a single cruise missile mission. Based on the results of this manuscript, we have developed an algorithm to create a complete mission plan (see [6]).

1.1 Description of the Problem

For this investigation, the threat regions are modelled as convex polygonal sets or circular regions. The mission plan is composed of a set of connected line segments, and adjacent line segments have a restriction on the maximum turn angle. The mission illustrated in Figure 1 is composed of three line segments (AB, BC, CD) and has two waypoints (B, C). The turn angles are given by $\angle EBC$ and $\angle FCD$. Mathematically, the problem of finding a feasible mission plan may be stated as follows:

> Given two points M and $T \in R^2$, a turn angle restriction of Z,
>
> and K convex sets $R_1, \ldots, R_K$, find a set of line segments
>
> $[Y_1, Y_2]$, $[Y_2, Y_3]$, ..., $[Y_S, Y_{S+1}]$ with $Y_1 = M$ and $Y_{S+1} = T$

such that no line segment intersects the interior of any $R_1, \ldots, R_K$ and no turn angle exceeds Z.

Figure 1 about here

Let $\Gamma(M, T, Z, R_1, \ldots, R_K)$ or simply Γ denote the set of feasible missions for a given missile. Note that if $\Gamma \neq \Phi$, then there are an infinite number of paths in Γ. If we could identify the objective function for mission planning, say f(y) for $y \in \Gamma$, then we would have a well-defined optimization problem. Unfortunately, the objective function for this problem involves some unknown trade-off between the length of the path from M to T and the number of segments needed for the mission. For the problem illustrated in Figure 2, the shortest path from M to T which does not pass through the threat is the line segment MA, the arc AB, and the line segment BT. Approximating this path with line segments, as illustrated in Figure 2b results in an infinite number of such segments. Clearly, mission planners are not interested in plans of the type illustrated in Figure 2b. For the problem illustrated in Figure 3, a two segment solution is given by MC, CT. However, the three

segment solution MA, AB, BT is shorter.

Figures 2 and 3 about here

Unlike other optimization problems, the mission planning problem as we describe it has no mathematical objective function. The objective is to find short feasible missions with only a few segments. Given a feasible mission plan, we believe that a mission planner can look at a visual display of the mission and conclude whether or not this is a good plan. Therefore, we are seeking algorithms which find feasible missions with three essential characteristics: a short total mission distance, a small number of segments, and turn angles within reasonable limits.

1.2 Survey of the Literature

Most research investigations in the area of mission planning begin by placing a grid over the combat theatre and applying a modification of Dijkstra's algorithm (see [2]) to obtain a safe path from the origin to the destination (see [1, 5, 16, 17, 27, 28]). There are two disadvantages to this approach. The

grid graph can become very large and consequently the algorithms can be very time-consuming. Also, the resulting paths may involve numerous line segments, some of which may have 90° (excessive) turn angles. This approach uses discrete methods for a problem that is fundamentally continuous. We believe that continuous methods that consider the geometry of the problem are preferable to discrete methods.

1.3 Related Algorithms

The robotics literature on path planning in the presence of obstacles is related to the mission planning problem. Two general classes of problems are addressed, one related to mobile robots and the other related to industrial robotic arms (manipulators).

Efficient methods for generating shortest paths for mobile robots have been developed. They are generally based on finding the shortest path in the tangent graph (or visibility graph) consisting of the collison-free tangents between all obstacles. For convex polygonal obstacles, this method is based on the premise that when two points in a plane are not visible from each other, the shortest path always contains one or more vertices of the obstacle. Part of

a tangent graph is illustrated in Figure 4. The mobile robot will move from M to T on the line segments composed of edges of the polygons and edges linking extreme points of the polygons. This approach works best for polygons and does not easily adapt to obstacles having curved surfaces. Although the circles may be approximated by polygonal shapes, a close approximation requires many turns. In the case of mission planning, a path having many segments is undesirable. Variations of this strategy can be found in [7, 10, 11, 12, 13].

Others have treated this mobile robot problem as a control problem. Shortest paths are generated by following the negative gradient of some return function. Multiple obstacles are avoided one at a time and the paths are smooth curves (see [18, 19, 26]).

Figure 4 about here

Operators of industrial robots have to either teach or program a manipulator so that it can function in the work space without collisions. In this case, the strategies for path planning include optimization which treats the

obstacle as a repulsive force field. When a robot arm moves close to the obstacle, a large repulsive force is generated which penalizes movement close to an obstacle. Optimization-based strategies may be found in [3, 4]. Other strategies include sensor-based motion planning [21, 22], control methods [15], and geometric methods [24]. An algorithm for a coordinate measuring machine which requires that a probe be positioned at various locations on a part may be found in [14]. These algorithms generally produce curved paths and also involve considerations of velocity, acceleration, joint types, and manipulator range.

The problem of finding shortest paths in 2D with obstacles has been addressed by Zheng and Iyengar [25]. They place a uniform grid over the plane and apply algorithms to the corresponding graph problem. For the 3D problem, Jun and Shin [9] divide the workspace into cells so that the problem becomes discrete with the obstacles occupying known cells. Using this idea, the problem is to find a sequence of neighboring cells from the origin to the destination which exclude all cells intersecting obstacles. These methods tend to find solutions that only allow certain angles between segments, such as 90° angles. Often such paths can be shortened, with fewer segments, if there is less restriction in the choice of turn angle.

The advantage of our algorithm is that it accomodates both circular and polygonal shapes. In addition, it generates straight line segments even for the circular shapes, which is required in order to program the missile waypoints. The small number of segments simplifies this programming, and the short paths are efficient and increase the probability of success.

1.4 Other Applications

Layout and routing problems also involve finding paths which avoid obstacles. Consider the problem of routing fiber optic cable between major cities. The objective is to determine a short route that avoids certain regions such as other cities, airports, military bases, lakes, and mountains. A similar problem is faced by public utilities (gas, water, electric), as well as highway planners. Trogneux, Doquet, and Mallet [20] address the problem of designing a high voltage transmission network which avoids obstacles such as airfields and army bases and takes environmental obstacles into account. Microwave communication links have a line-of-sight requirement. Finding the placement of microwave towers between major cities may be viewed in this way. In these types of applications, the turn angle constraint which our

algorithm addresses becomes unimportant.

1.5 Objective of the Investigation

Advances in graphics technology has led to a new field of study known as *scientific visualization* or just *visualization.* Research groups in physics, fluid dynamics, meteorology, medicine, molecular biology, and operations research are using visualization techniques (including animation) to help in problem-solving ([8]). In optimization applications, visualization is frequently used to display solutions to complex multidimensional problems. This investigation not only uses visualization to display the solutions, but also uses visualization to evaluate the solutions. Simply stated, the objective of this investigation is to develop algorithms which automatically generate feasible missions for cruise missiles which will be acceptable to mission planners. The algorithm provides paths which to human perception are a good balance betwen short total path distance and a small number of segments. Hence, the empirical analysis results in a set of figures in which appearance is the only criteria for evaluation.

2 CIRCULAR THREATS

In this section, the K convex sets $R_1, \ldots, R_K$ are assumed to be circles each with center $X_1, \ldots, X_K \in R^2$ and radii $r_1, \ldots, r_K$. The method described in this section is called the *circumscribed triangle algorithm.* It starts at M (the missile location) and constructs line segments by moving from a given segment point tangentially to the circumference of nearby threat regions, until a suitable straight-line path from a segment point to T (the target location) is found. This basic strategy is embedded in a branch-and-bound framework, so that a short path of this restricted type is obtained.

In this algorithm we allow the missile to intersect a point on the circumference of a threat region but do not allow the missile to fly through any interior point of a threat region. Given segments $[Y_{i-1}, Y_i]$ and $[Y_i, Y_{i+1}]$, the turn angle at segment point Y_i is defined to be the complement of angle $Y_{i-1}Y_iY_{i+1}$.

Let $d(E, F)$ denote the Euclidean distance between points E and F. The basic step in proceeding from segment point Y_i tangentially to a (nearby) threat region $R_k = \{s : d(X_k, s) \leq r_k\}$ involves constructing the circumscribing triangle on R_k having one vertex at Y_i (see Figure 5).

Figure 5 about here

The tangential points in Figure 5 are T_1 , T_2, and T_3 and the vertex points are Y_i, V_1, and V_2. Sides Y_iV_1 and Y_iV_2 are the equal sides of the (isosceles) circumscribing triangle.

The tangential points in Figure 6 are T_1 , T_2, and T_3 and the vertex points are Y_i, V_1, and V_2. Sides Y_iV_1 and Y_iV_2 are the equal sides of the (isosceles) circumscribing triangle. Tangential point T_3 is easily computed since it lies on the line through Y_i and X_k with $d(X_k, T_3) = r_k$. Auxillary point Z formed by dropping a normal to Y_iT_3 from T_1 (or T_2) can be used to obtain T_1 and V_1 (or T_2 and V_2). Note that $\triangle Y_iZT_1$, $\triangle Y_iT_3V_1$, $\triangle Y_iT_1X_k$, and $\triangle ZT_1X_k$ are all similar triangles.

Since $\triangle ZT_1X_k \sim \triangle Y_iT_1X_k$,

$$\frac{X_kZ}{X_kT_1} = \frac{X_kT_1}{X_kY_i} \quad \text{or} \quad \frac{d(X_k, Z)}{r_k} = \frac{r_k}{d(Y_i, X_k)}$$

so that

$$d(X_k, Z) = (r_k)^2/d(Y_i, X_k) \ .$$

Also since $\triangle ZT_1X_k \sim \triangle Y_iT_1X_k$,

$$\frac{ZT_1}{X_kT_1} = \frac{T_1Y_i}{X_kY_i} \quad \text{or} \quad \frac{d(T_1,Z)}{r_k} = \frac{\sqrt{d(Y_i,X_k)^2-(r_k)^2}}{d(Y_i,X_k)}$$

so that

$$d(T_1,Z) = r_k\sqrt{d(Y_i,X_k)^2-(r_k)^2}/d(Y_i,X_k) \ .$$

Since $\triangle Y_iT_3V_1 \sim \triangle Y_iZT_1$,

$$\frac{V_1T_3}{Y_iT_3} = \frac{T_1Z}{Y_iZ} \quad \text{or} \quad \frac{d(T_3,V_1)}{d(Y_i,X_k)+r_k} = \frac{d(T_1,Z)}{d(Y_i,X_k)-d(X_k,Z)}$$

so that

$$\begin{aligned} d(T_3,V_1) &= \frac{(d(Y_i,X_k)+r_k)r_k\sqrt{d(Y_i,X_k)^2-(r_k)^2}}{d(Y_i,X_k)(d(Y_i,X_k)-d(X_k,Z))} \\ &= \frac{(d(Y_i,X_k)+r_k)r_k\sqrt{d(Y_i,X_k)^2-(r_k)^2}}{d(Y_i,X_k)(d(Y_i,X_k)-(r_k)^2/d(Y_i,X_k))} \\ &= \frac{d(Y_i,X_k)r_k)+(r_k)^2}{\sqrt{d(Y_i,X_k)^2-(r_k)^2}} \ . \end{aligned}$$

Figure 6 about here

Thus points V_1 and V_2 can be readily obtained by moving a distance $d(T_3, V_1)$ along the two normals to Y_1T_3 at point T_3. The points T_1 and T_2 can be obtained by moving a distance $\sqrt{d(Y_i, X_k)^2 - (r_k)^2}$ from Y_i toward V_1 and V_2, respectively, or by computing the nearest points to X_k lying on the lines through Y_i and V_1 or V_2.

We intend to make use of these tangential lines to construct the next segment, if possible. Before using points on either tangential line, we first make sure that the respective turn angles at Y_i would be within the turn angle limit.

The points V_1 and V_2 have a special property. Each is the first point on the tangential line from Y_i through itself which has a complete 180° field of view (ignoring turn angle constraints) of the side of R_k opposite Y_i. By contrast, tangent points T_1 and T_2 each have less than a 90° field of view (ignoring turn angle constraints) of that side (see Figure 7). With this in mind V_1 (or V_2) would seem to be a good choice for the end point of a next segment in our path construction. We actually attempt to use a point between T_1 and V_1 (or between T_2 and V_2) for the next end point. In general, such a point can be represented as a convex combination $Y_{i+1} = (1-\alpha)T_1 + \alpha V_1$ (or $Y_{i+1} = (1-\alpha)T_2 + \alpha V_2$), for some $0 \leq \alpha \leq 1$. In order for such a point

Y_{i+1} to be usable, we must insure that the segment $[Y_i, Y_{i+1}]$ is clear. (A line segment is defined to be *clear* if it does not intersect the interior of any of the threat regions.)

Figure 7 about here

We first search for a minimum value of α with $0 \leq \alpha \leq 1$ such that $[Y_i, (1-\alpha)T_1 + \alpha V_1]$ (or $[Y_i, (1-\alpha)T_2 + \alpha V_2]$) is clear and $[(1-\alpha)T_1 + \alpha V_1, T]$ (or $[(1-\alpha)T_2 + \alpha V_2, T]$) is clear without exceeding the turn angle limit at $(1-\alpha)T_1 + \alpha V_1$ (or $(1-\alpha)T_2 + \alpha V_2$). If such a point is found we have a *feasible completion* by adding the two segments $[Y_i, Y_{i+1} = (1-\alpha)T_1 + \alpha V_1]$ and $[Y_{i+1} = (1-\alpha)T_1 + \alpha V_1, Y_{i+2} = T]$ (or $[Y_i, Y_{i+1} = (1-\alpha)T_2 + \alpha V_2]$ and $[Y_{i+1} = (1-\alpha)T_2 + \alpha V_2, Y_{i+2} = T]$).

If such a point leading to a feasible completion cannot be found, it would seem natural to use V_1 (or V_2) by default, due to its wide field of view. However, in practice this leads to longer paths and we find it better to use a point between T_1 and V_1 (or between T_2 and V_2) for the next end point. We have parameterized this by defining a *backoff percentage* β which specifies

that, in this case, Y_{i+1} is to be chosen as the convex combination $(1-\beta)T_1 + \beta V_1$ (or $(1-\beta)T_2 + \beta V_2$). Experimentally, we have found $\beta = \frac{1}{2}$ to be a good choice.

The branch-and-bound algorithm we have developed based on this circumscribed triangle approach is given as follows:

Algorithm : *circumscribe*

Inputs :

M	- missile location
T	- target location
K	- number of threats
$X_1, ..., X_K$	- center points of the threats
$r_1, ..., r_K$	- radii of the threats
Z	- turn angle limit (degrees)
β	- fall back parameter $(0 < \beta \leq 1)$

Outputs :

D	- length of best constructed solution path
S	- number of segments in best constructed solution path

$L_1, ..., L_S$ - segments of best constructed solution path

Constructs :

B - pointer to last point in best constructed solution path

Q - branch-and-bound queue

$d(E, F)$ - distance from E to F

$L(E, F)$ - line segment from E to F

$R(C, r)$ - threat region with center C and radius r

$S(C, r)$ - interior of threat region with center C and radius r

I - number of points constructed

Y_i - i^{th} constructed point

B_i - pointer to point previous to Y_i

D_i - path distance prior to Y_i

S_i - path segments prior to Y_i

E_i - set of threat points eligible for moving away from point Y_i

T_1, T_2, T_3 - constructed tangent points of circumscribled triangle

V_1, V_2 - constructed vertices of circumscribled triangle

begin

$D \leftarrow \infty$; $S \leftarrow 0$; $B \leftarrow 0$;

if $L(M,T) \cap S(X_i, r_i) = \Phi$ for $i = 1..K$ **then**

$D \leftarrow d(M,T)$; $S \leftarrow 1$; $L_1 \leftarrow [M,T]$;

else

begin

$I \leftarrow 1$; $Y_1 \leftarrow M$; $B_1 \leftarrow 0$; $D_1 \leftarrow 0$; $S_1 \leftarrow 0$; $E_1 \leftarrow \{R_1, ..., R_K\}$;

put $(Y_1, B_1, D_1, S_1, E_1)$ on Q;

while $Q \neq \Phi$ **do**

begin

remove some $(Y_i, B_i, D_i, S_i, E_i)$ from Q;

$b \leftarrow B_i$;

select some R_j from E_i; $E_i \leftarrow E_i \setminus R_j$;

if $E_i \neq \Phi$ **then** put $(Y_i, B_i, D_i, S_i, E_i)$ back on Q;

form the triangle $Y_i V_1 V_2$ circumscribing R_j having

its tangent points $T_1 \in Y_i V_1, T_2 \in Y_i V_2, T_3 \in V_1 V_2$;

for $k = 1$ **to** 2 **do**

begin

if $180 - Y_b Y_i T_k \leq Z$ and

$L(Y_i, T_k) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$ and

$D_i + d(Y_i, T_k) + d(T_k, T) < D$ **then**

begin

search for a point U on $L(T_k, V_k)$ nearest to T_k satisfying:

$180 - Y_i U T \leq Z$ and

$L(Y_i, U) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$ and

$L(U, T) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$;

if such a point U exists **then**

begin

if $D_i + d(Y_i, U) + d(U, T) < D$ **then**

begin

$I \leftarrow I + 1$;

$Y_I \leftarrow U$; $B_I \leftarrow i$; $D_I \leftarrow D_i + d(Y_i, U)$; $S_I \leftarrow S_i + 1$;

$m \leftarrow I$; $I \leftarrow I + 1$;

$Y_I \leftarrow T$; $B_I \leftarrow m$; $D_I \leftarrow D_m + d(U, T)$; $S_I \leftarrow S_i + 2$;

$D \leftarrow D_I;\ \ S \leftarrow S_I;\ \ B \leftarrow I;$

end;

else

begin

if $E_i \neq \Phi$ **then**

begin

$U \leftarrow (1-\beta)T_k + \beta V_k;$

if $L(Y_i, U) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$ **then**

begin

$I \leftarrow I + 1;$

$Y_I \leftarrow U; B_I \leftarrow i; D_I \leftarrow D_i + d(Y_i, U); S_I \leftarrow S_i + 1; E_I \leftarrow E_i;$

put $(Y_I, B_I, D_I, S_I, E_I)$ on Q;

end;

end;

end;

end;

end;

end;

$i \leftarrow S;\ \ k \leftarrow B;$

while $k \neq 0$ **do** $\ \ j \leftarrow B_k;\ \ L_i \leftarrow [Y_j, Y_k];\ \ i \leftarrow i - 1;\ \ k \leftarrow j;$

end;

Figure 8 shows an optimal path through a region containing three threats generated by the circumscribed triangle algorithm. Figure 9 shows additional partial paths produced by the branch-and-bound process contained in the algorithm. The actual implementation, whose results are reported in Appendix A, actually makes two applications of the algorithm, the second of which has the roles of missile and target reversed. Of course, any path obtained with missile and target reversed would itself need to be reversed. The shortest of the two paths generated is selected as the best path.

Figures 8 and 9 about here

---

3 CIRCULAR THREATS AND A PROBABILITY CONSTRAINT

In this section, a threat is represented by a circle with a given center X and a given radius r. Let Q denote the point along the mission plan at which the missile is nearest X. If $||Q - X|| > r$, then the threat at X is too far away to harm the missile and the probability of a successful traversal is defined to be 1. Otherwise, some function $\mathrm{p}(Q, X, r)$ is used to determine the probability of a successful traversal. When there are multiple threats, the probability of a successful traversal is the product of the probabilities along each segment, taking into account all of the threats.

The problem of finding a feasible mission plan which incorporates a probability constraint, can be defined mathematically as follows:

> Given a target probability P, two points M and $T \in R^2$, and K circles, with centers $X_1, ..., X_K \in R^2$ and radii $r_1, ..., r_K$, find a set of line segments $[Y_1, Y_2]$, $[Y_2, Y_3]$,..., $[Y_S, Y_{S+1}]$, with $Y_1 = M$ and $Y_{S+1} = T$, such that the probability of a successful traversal is at least P.

The quality of a mission plan is measured by the length of the path and the

number of path segments. The best path is the straight line from M to T. If this is not feasible (i.e. the probability of a succcessful traversal is less than P), then we seek a short path having only a few segments.

The probabilistic incursion triangle algorithm allows the missile to fly through the interior of a threat region and employs a probability model to compute the probability of survival for a path from the missile location M to the target location T which consists of line segments which may either be tangent to a circular threat region or cut through the region (an incursion).

The algorithm operates within the same branch-and-bound algorithmic structure used in the circumscribed triangle algorithm. The major differences are (1) when attempting to construct a new segment by approaching a threat region from a previous segment endpoint, several equilateral triangles passing through the threat region in addition to the circumscribed triangle may be considered, and for each such triangle, potentially two points are added to the branch-and-bound queue, and (2) in searching for a final segment linking a previous segment endpoint to the target, that segment may intersect threat regions (with a consequent reduction in survival probability).

For threat region k with center X_k and radius r_k, the probability model assumes that the probability of survival of a line segment approaching a

threat region is a function only of the minimum distance from the segment to X_k. Further, the model assumes independence so that the overall survival probability for a path is the product of the survival probabilities for each segment in the path.

For developmental purposes we have hypothesized a simple probability function which is a quadratic of the form

$$Pr(d) \;=\; ad^2 + c \;,$$

where for threat region k,

$$Pr(r_k) = .99 \text{ and } Pr(0) = .50 \;.$$

Figure 10 illustrates a circumscribed triangle YiV_1V_2 and an incursion triangle $Y_iQ_1Q_2$ for a common threat region R_k. In the circumscribed triangle algorithm, with respect to the segments Y_iV_1 and Y_iV_2, the line segments T_1V_1 and T_2V_2 are searched first for a clear completion segment to the target, before the points V_1 and V_2 are added to the branch-and-bound queue.

This also occurs in the probabilistic incursion triangle algorithm, but the completions can intersect threat regions if the total survival probability is at least P and the segments Y_iV_1 and Y_iV_2 will each have survival probability

.99. Also, in the probabilistic incursion triangle algorithm, with respect to the segments Y_iQ_1 and Y_iQ_2, the line segments P_1Q_1 and P_2Q_2 are searched first for a completion to the target with total survival probability at least P, before the points V_1 and V_2 are added to the branch-and-bound queue.

Figure 10 about here

The number of incursion triangles considered in approaching a threat region from a previous segment endpoint depends on the accumulated number of segments and an adjustable parameter $\hat{S}$ which is roughly interpreted as the expected number of segments in an optimal path. In the testing described in Appendix B, $\hat{S}$ was set at 3. If S_i is the number of segments in a partial path from the missile location to segment endpoint Y_i and P_i is the survival probability of this partial path, incursion triangles with individual segment survival probabilities of

$$.99(P/P_i)\ ,\ .99\left(\frac{P/P_i}{2}\right)\ ,\ \ldots\ ,\ .99\left(\frac{P/P_i}{\hat{S}-S}\right)$$

will be considered, unless these probabilities are very close to .99.

The twenty test problems presented in Appendix A were solved using $P = 90\%$, $P = 75\%$, and $P = 60\%$, and the results are summarized in Table 1. The deterministic algorithm is the circumscribed triangle algorithm. Note that setting $P = 60\%$ subsantially reduces the number of waypoints, but only reduces the total distance by 4%. The paths obtained by the deterministic algorithm and the new algorithm with $P = 60\%$ may be found in Appendix B. For most problems, the two paths are similar, with the new algorithm taking short cuts through one or more threats. Problem 7 illustrates a case where the new algorithm selects a completely different path. For this problem, the deterministic algorithm produces a path in which all threats are above the path, while the new algorithm uses a mixed strategy. The deterministic strategy is unable to use a single straight line for any of these test problems, whereas the probabilistic algorithm finds this to be best for several of them.

Table 1 about here

---

4 POLYGONAL THREATS

In this section, we extend the ideas presented in Section 2 to handle the case of threat regions modelled as convex hulls. The algorithm described below is fundamentally the same as the algorithm described in Section 2. The modified algorithm is given in Appendix C. The original algorithm required that all threat regions be modelled as circular regions. The major modification we have made also allows a threat region to be modelled as the convex hull of a given set of points.

The original and modified algorithms start at M (the missile location) and construct line segments by moving from a given segment point tangentially to the boundary of a nearby threat region. This process continues until a suitable straight-line path from a segment point to T (the target location) is found. This basic strategy is embedded in a branch-and-bound framework, so that a short path of this restricted type is obtained.

The missile is allowed to intersect a point on the boundary of a threat region but is not allowed to fly through any interior point of a threat region. Further, a turn angle limit is enforced at each segment point, since a missile is unlikely to be making a sharp turn at such a point. Given segments

$[Y_{i-1}, Y_i]$ and $[Y_i, Y_{i+1}]$, the turn angle at segment point Y_i is defined to be $180° - \angle Y_{i-1}Y_iY_{i+1}$.

Let $d(E, F)$ denote the Euclidean distance between points E and F. The basic step in proceeding from segment point Y_i tangentially to a (nearby) threat region R_k involves constructing the circumscribing triangle for R_k which has one vertex at Y_i. Previously, a procedure was described for obtaining the circumscribing triangle used when moving from a segment point Y_i tangentially past a circular threat region defined by a center point X_k and radius r_k.

The modified algorithm can also obtain the circumscribing triangle to be used in moving from a segment point Y_i past a threat region R_k defined as the convex hull of a set of generator points $\{G_1, \ldots, G_l\}$ (see Figure 11).

Figure 11 about here

The generator points defining the circumscribed triangle in Figure 11 are the side points T_1 , T_2, and T_3 (which play a role similar to that of the

tangential points in the case of a circular region) and the vertex points are Y_i, V_1, and V_2. Sides Y_iV_1 and Y_iV_2 are the equal sides of the (isosceles) circumscribing triangle. Points T_1 and T_2 are the generator points associated with the largest angle on distinct generator points having vertex Y_i. This angle is actually determined by computing

$$\min_{a \neq b} \frac{(G_a - Y_i) \cdot (G_b - Y_i)}{| (G_a - Y_i) || (G_b - Y_i) |} .$$

The generator point T_3 is associated with the largest inner product formed from generator points and (normal vector)

$$\left(\frac{(T_1 - Y_i)}{| (T_1 - Y_i) |} + \frac{(T_2 - Y_i)}{| (T_2 - Y_i) |} \right) .$$

Note that the line from Y_i in the direction given by the above normal vector bisects the angle $T_1Y_iT_2$ and determines the auxillary point Z, which in turn bisects the segment V_1V_2. The maximum inner product is computed as

$$\eta = \max_a G_a \cdot \left(\frac{(T_1 - Y_i)}{| (T_1 - Y_i) |} + \frac{(T_2 - Y_i)}{| (T_2 - Y_i) |} \right) .$$

Points V_1 and V_2 are then given by

$$V_1 = Y_i + \alpha_1 \left(\frac{(T_1 - Y_i)}{| (T_1 - Y_i) |} \right)$$

$$V_2 \;=\; Y_i \;+\; \alpha_2 \left(\frac{(T_2 - Y_i)}{|\,(T_2 - Y_i)\,|} \right)$$

where α_1 and α_2 are determined from the solution of the equations:

$$\eta \;=\; \left(Y_i \;+\; \alpha_1 \left(\frac{(T_1 - Y_i)}{|\,(T_1 - Y_i)\,|} \right) \right) \cdot \left(\frac{(T_1 - Y_i)}{|\,(T_1 - Y_i)\,|} + \frac{(T_2 - Y_i)}{|\,(T_2 - Y_i)\,|} \right)$$

$$\eta \;=\; \left(Y_i \;+\; \alpha_2 \left(\frac{(T_2 - Y_i)}{|\,(T_2 - Y_i)\,|} \right) \right) \cdot \left(\frac{(T_1 - Y_i)}{|\,(T_1 - Y_i)\,|} + \frac{(T_2 - Y_i)}{|\,(T_2 - Y_i)\,|} \right) \;,$$

which place V_1 and V_2 on the line normal to Y_iZ at Z. Note that T_3 is also on that normal line.

Let W_1 (or W_2) be a fixed point along the line segment starting at T_1 (or T_2) through V_1 (or V_2). The algorithm searches for a minimum value of α with $0 \leq \alpha \leq 1$ such that $[Y_i, (1-\alpha)T_1 + \alpha W_1]$ (or $[Y_i, (1-\alpha)T_2 + \alpha W_2]$) does not intersect the interior of any of the threat regions, $[(1-\alpha)T_1 + \alpha W_1, T]$ (or $[(1-\alpha)T_2 + \alpha W_2, T]$) does not intersect the interior of any of the threat regions, and the turn angle limit was not exceeded at $(1-\alpha)T_1 + \alpha W_1$ (or $(1-\alpha)T_2 + \alpha W_2$). In the original algorithm, W_1 was V_1 and W_2 was V_2. The modified algorithm performs this search for both a circular threat region

and a convex hull threat region, and if the line segment from Y_i through T_1 (or T_2) to the boundary does not interesect any of the threat regions, then W_1 (or W_2) is taken to be that boundary point instead. If such a suitable point is found, a *feasible completion* is formed by adding the two segments $[Y_i, Y_{i+1} = (1-\alpha)T_1 + \alpha W_1]$ and $[Y_{i+1} = (1-\alpha)T_1 + \alpha W_1, Y_{i+2} = T]$ (or $[Y_i, Y_{i+1} = (1-\alpha)T_2 + \alpha W_2]$ and $[Y_{i+1} = (1-\alpha)T_2 + \alpha W_2, Y_{i+2} = T]$) to the line segments already constructed leading from M to Y_i.

If such a point leading to a feasible completion cannot be found, a point between T_1 and V_1 (or between T_2 and V_2) is used for the next end point. In the original algorithm, a *backoff percentage* $\beta = 50\%$ was used so that the next segment point Y_{i+1} is the convex combination $(1-\beta)T_1 + \beta V_1$ (or $(1-\beta)T_2 + \beta V_2$). In the modified algorithm, β is computed so that $(1-\beta)T_1 + \beta V_1$ (or $(1-\beta)T_2 + \beta V_2$) is the first point from T_1 toward V_1 (or from T_2 toward V_2) such that a line at that point with the limiting turn angle intersects the boundary of the threat region, but not the interior. (For a convex hull threat region, a degenerate case is possible in which $T_1 = V_1$ or $T_2 = V_2$ and a distinct intermediate point is neither possible nor needed.)

As with the original implementation, the algorithm is applied twice, the second time with the missile and target locations reversed, and the better of

the two paths generated is taken as the recommended path.

The real test for this algorithm is whether or not it develops paths which look reasonable when displayed on a two-dimensional drawing. The paths for twenty problems are displayed in Appendix D. These paths not only appear reasonable, but also appear to be of minimum length. In addition, the solution time on a Dec 5000 workstation was less than one minute for each of these test problems.

5 SUMMARY AND CONCLUSIONS

This manuscript presents a new algorithm to automatically generate a path composed of line segments from a given origin (missile location) to a given destination (target location) which avoids obstacles (threat regions) modelled as circles. The algorithm exploits the geometry of the problem description and mimics the trial-and-error strategy a human mission planner might wish to apply to this graphical problem. The basic idea is that short paths which avoid circular obstacles should coincide with the edges of a circumscribing triangle defined by a current point and a circle between the current point and the target. A heuristic strategy is applied to determine the stopping point

along each of these two sides. Ideally the stopping point will have a clear path to the target, in which case a path has been discovered. If this is not the case, then this is repeated with each of the previous stopping points. At each step there are two edges which can be followed to obtain a path around an obstacle. All such paths are examined and the shortest is retained. The algorithm reverses the roles of the missile and the target and the shortest path from each application is the path reported by the algorithm.

Two extensions of this basic algorithm are also presented in this manuscript. One extension assigns a probability of a safe traversal for each leg of a path from the origin to the destination. Legs are allowed to pass through the threat circles with a corresponding reduction in the probability of a safe traversal along this leg. The algorithm is extended to produce short paths which satisfy a given probability restriction. The second extension is for threat regions which are convex polygons. The mission planning problem as described in this manuscript has no objective function and can only be judged qualitatively by examination of a graphical display of the problem and solution path. We believe that the sixty displays presented in this manuscript provide a convincing case that our algorithms provide good solutions.

References

[1] A. Boroujerdi, C. Dong, Q. Ma, and B. Moret, "Joint Routing in Networks," undated technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM.

[2] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* 1 (1959) 269 - 271.

[3] M. Galicki, "Optimal Planning of a Collision-Free Trajectory of Redundant Manipulators," *The International Journal of Robotics Research*, 11, 6, (1992) 549-559.

[4] Z. Y. Guo and T. C. Hsia, "Joint Trajectory Generation for Redundant Robots in an Environment with Obstacles," *Journal of Robotic Systems*, 10, 2 (1993) 199-215.

[5] R. Helgason, J. Kennington, and K. Lewis, "Finding Safe Paths in Networks," Technical Report 96-CSE-7, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1996).

[6] R. Helgason, J. Kennington, and K. Lewis, "Cruise Missile Strike Planning: Automatic Multiple Path Generation," Technical Report 97-CSE-

24, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1997).

[7] K. Jiang, L. D. Seneviratne, and S. W. E. Earles, "Three-Dimensional Shortest Path Planning in the Presence of Polyhedral Obstacles," *Proceedings of the Institute of Mechanical Engineers: Journal of Mechanical Engineeing Science, Part C*, 210 (1996) 373-381.

[8] C. V. Jones, "Visualization and Optimization," *ORSA Journal on Computing*, 6, 3 (1994) 221-257.

[9] S. Jun and K. G. Shin, "Shortest Path Planning in Discretized Workspaces Using Dominance Relation," *IEEE Transactions on Robotics and Automation*, 7, 3 (1991) 342-350.

[10] J. K. Lee and S. M. Song, "Path Planning and Gait of Walking Machine in an Obstacle-Strewn Environment," *Journal of Robotic Systems*, 8, 6 (1991) 801-827.

[11] Y. H. Liu and S. Arimoto, "Path Planning Using a Tangent Graph for Mobile Robots Among Polygonal and Curved Obstacles," *The International Journal of Robotics Research*, 11, 4 (1992) 376-382.

[12] Y. H. Liu and S. Arimoto, "Computation of the Tangent Graph of Polygonal Obstacles by Moving-Line Processing," *IEEE Transactions on Robotics and Automation*, 10, 6 (1994) 823-830.

[13] Y. H. Liu and S. Arimoto, "Finding the Shortest Path of a Disk Among Polygonal Obstacles Using a Radius-Independent Graph," *IEEE Transactions on Robotics and Automation*, 11, 5 (1995) 682-691.

[14] E. Lu, J. Ni, and S. M. Wu, "An Algorithm for the Generation of an Optimum CMM Inspection Path," *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control*, 116 (1994) 396-401.

[15] S. K. Singh and M. C. Leu, "Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints," *The International Journal of Robotics Research*, 10, 2 (1991) 171-187.

[16] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Fast Computation of Optimal Paths Using a Parallel Dijkstra Algorithm with Embedded Constraints," *Neurocomputing* 8, (1995) 195 - 212.

[17] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Autorouting Using a Parallel Dijkstra Algorithm with Embedded Constraints," undated

technical report, The Naval Surface Warfare Center, Dahlgren, VA.

[18] I. Spangelo and O. Egeland, "Trajectory Planning and Collision Avoidance for Underwater Vehicles Using Optimal Control," *IEEE Journal of Oceanic Engineering*, 19, 4 (1994) 502-511.

[19] S. Sundar and Z. Shiller, "Optimal Obstacle Avoidance Based on the Hamilton-Jacobi-Bellman Equation," *IEEE Transactions on Robotics and Automation*, 13, 2 (1997) 305-310.

[20] F. Trogneux, M. Doquet, P. Mallet, "Taking Environmental Constraints into Account in the Planning of the Extra High Voltage Transmission Network; EDF's Approach," IEEE Transactions on Power Delivery, 11, 4 (1996), 1901 - 1906.

[21] C. S. Tseng and T. S. Lue, "Path Planning for Robot Manipulators in Polyhedral Objects Environment," *Journal of Robotic Systems*, 12,10 (1995) 637-646.

[22] D. Wang and Y. Hamam, "Optimal Trajectory Planning of Manipulators with Collision Detection and Avoidance," *The International Journal of Robotics Research*, 11, 5 (1992) 460-468.

[23] B. Welch, *Practical Programming in Tcl and Tk*, Prentice-Hall, Inc., Upper Saddle River, NJ (1995).

[24] T. H. Wu and K. Y. Young, "Path Planning in the Presence of Obstacles Based on Task Requirements," *Journal of Robotic Systems*, 11, 8 (1994) 703-716.

[25] S. Q. Zheng, J. S. Lim, and S. S. Iyengar, "Finding Obstacle-Avoiding Shortest Paths Using Implicit Connection Graphs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15, 1 (1996) 103-110.

[26] Q. Zhu, "Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation," *IEEE Transactions on Robotics and Automation*, 7, 3 (1991) 390-397.

[27] M. Zuniga and P. Gorman, "Threat Site Overflight Modeling for Strike Route Optimization," undated technical report, Naval Research Laboratory, Washington, D.C.

[28] M. Zuniga, J. Uhlmann, and J. Hofmann, "The Interdependent Joint Routing Problem: Description and Algorithmic Approach," undated

technical report, Naval Research Laboratory, Washington, D.C.

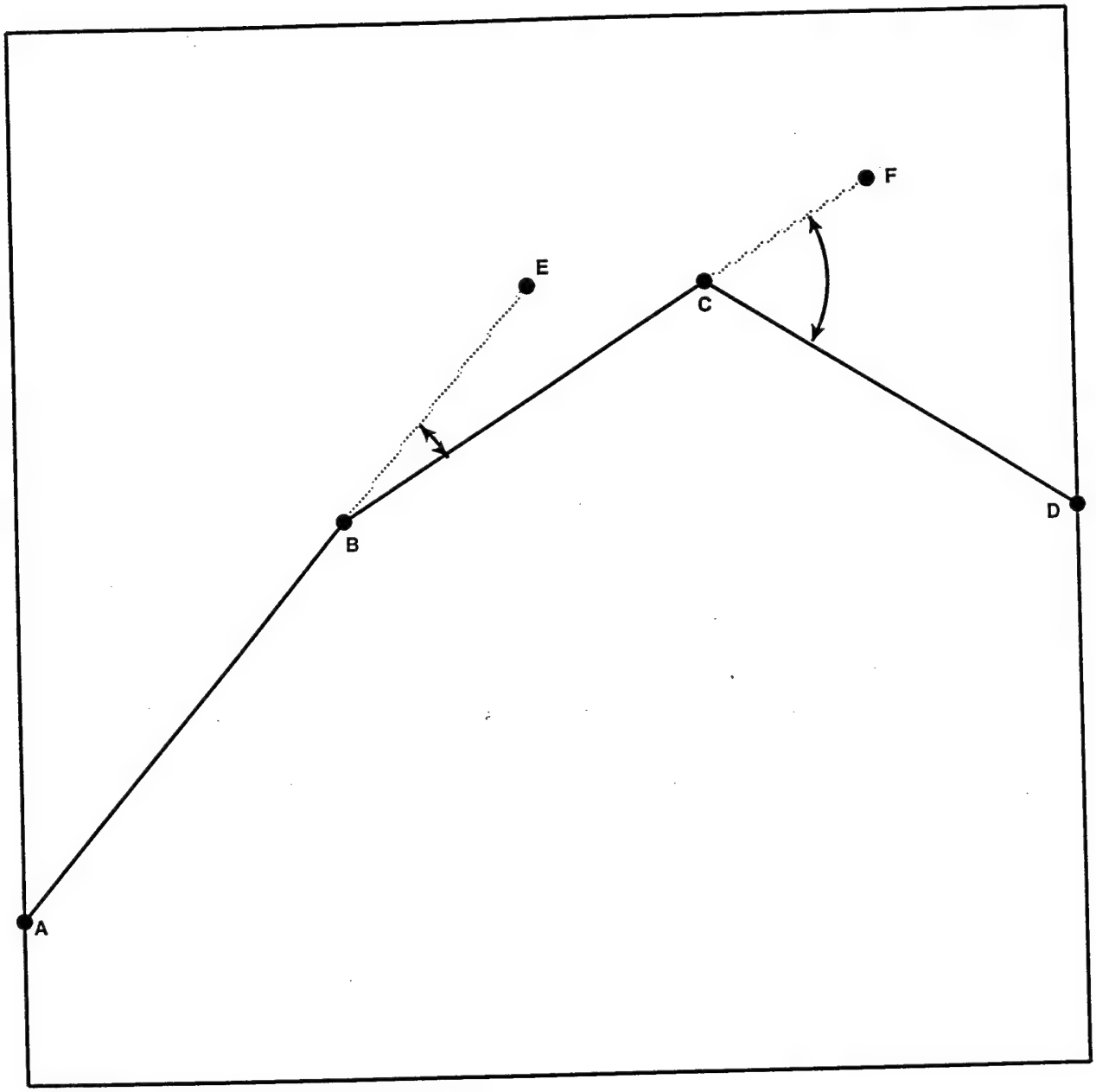
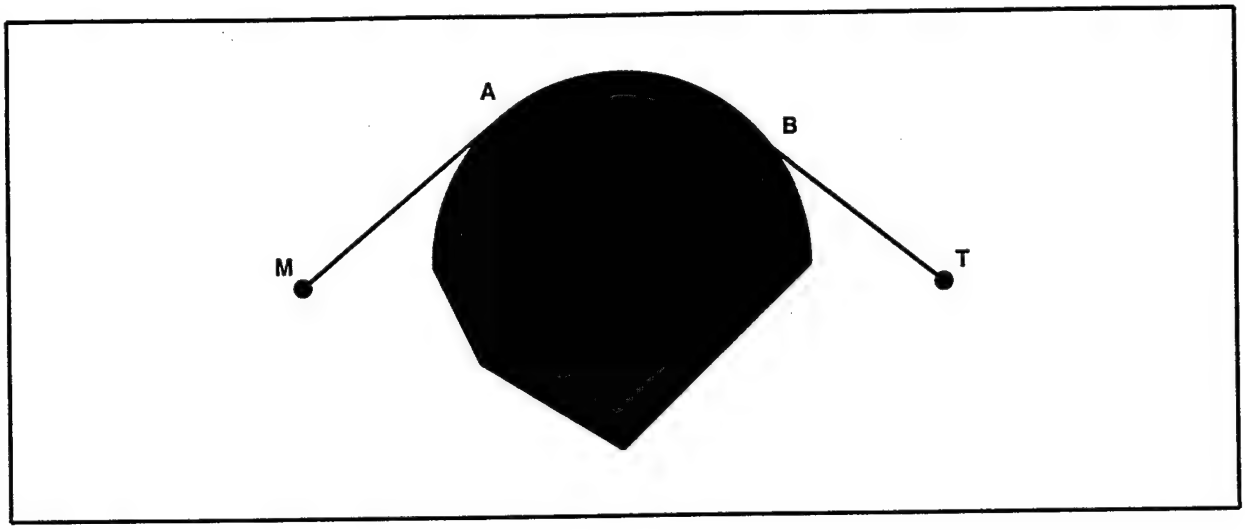
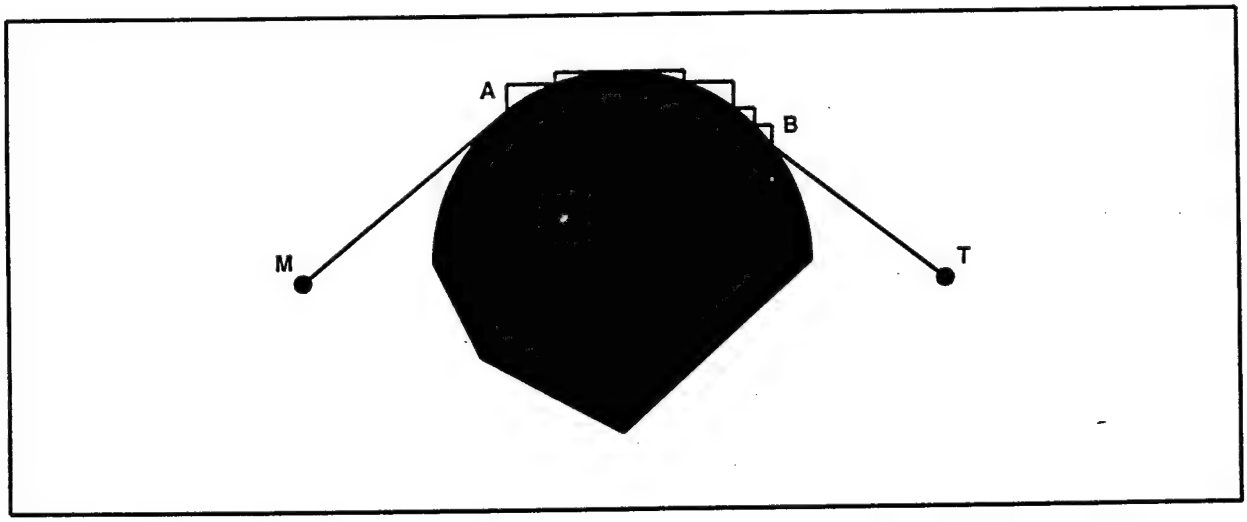


Figure 1: Illustration of Turn Angles



a. Shortest Path



b. Shortest Path Using Line Segments

Figure 2: A Mission with a Single Threat

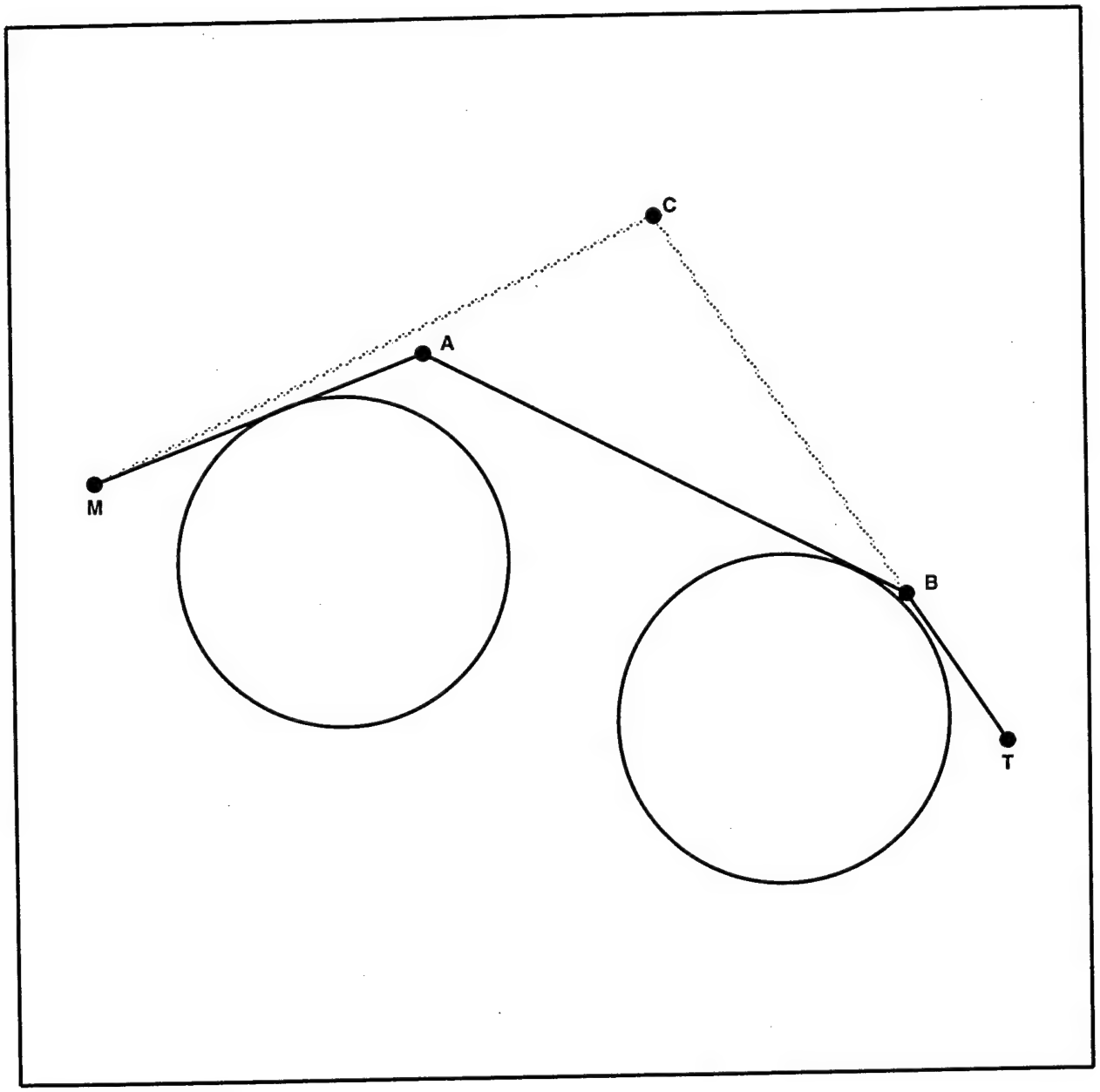


Figure 3: A Mission with Two Threats

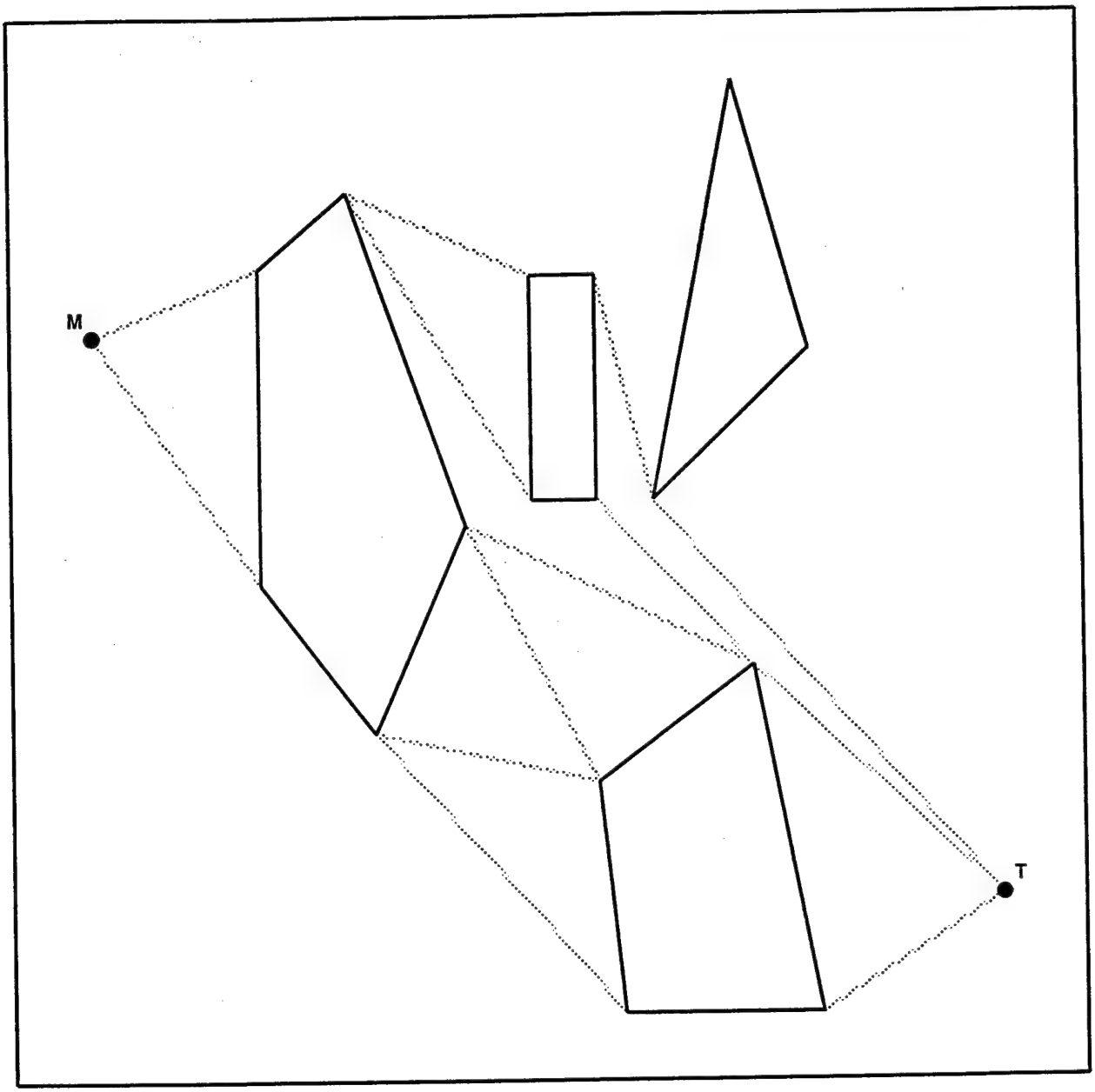


Figure 4: Partial Tangent Graph

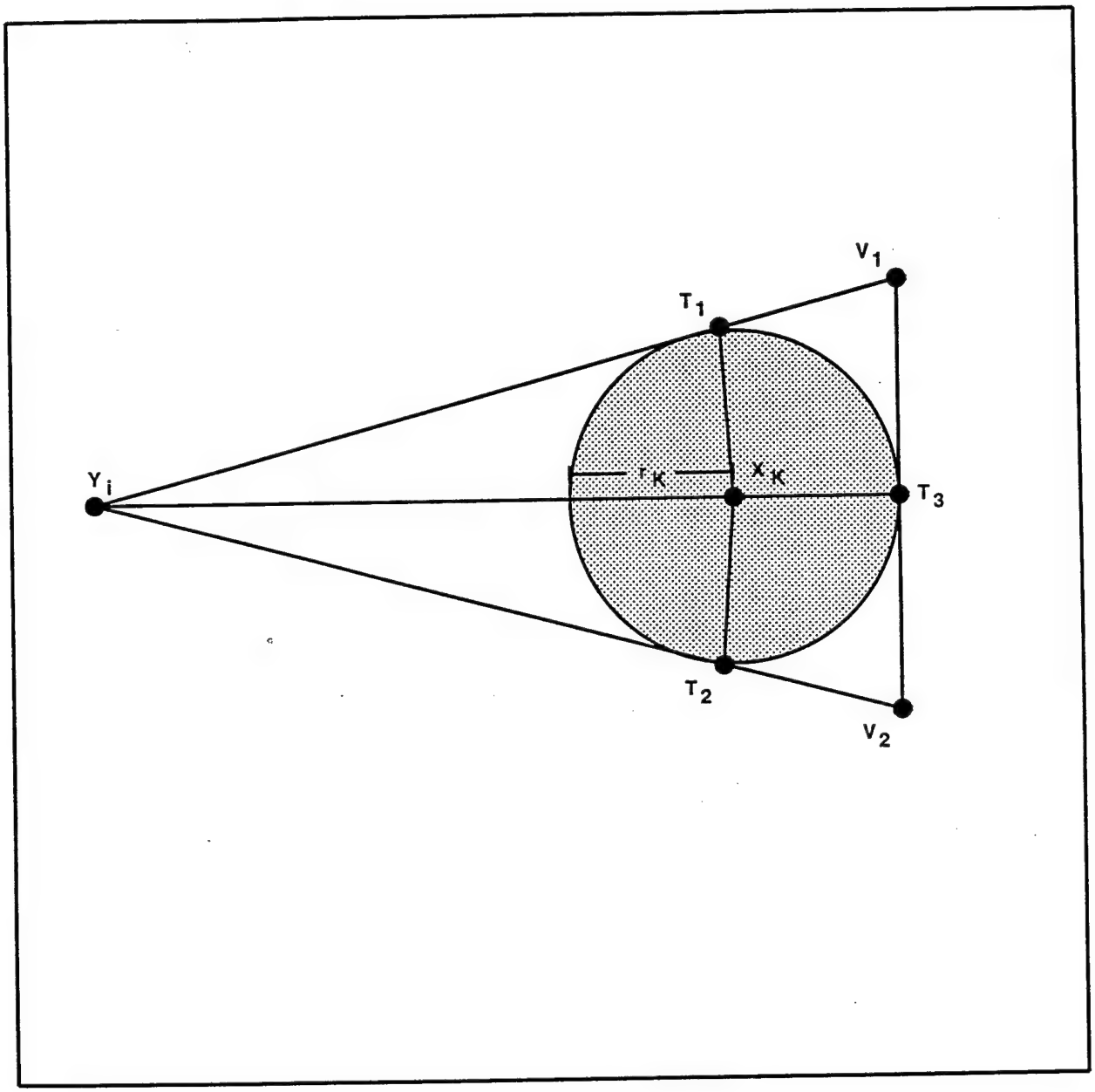


Figure 5: Circumscribing Triangle with Vetex Y_i for Threat Region R_K

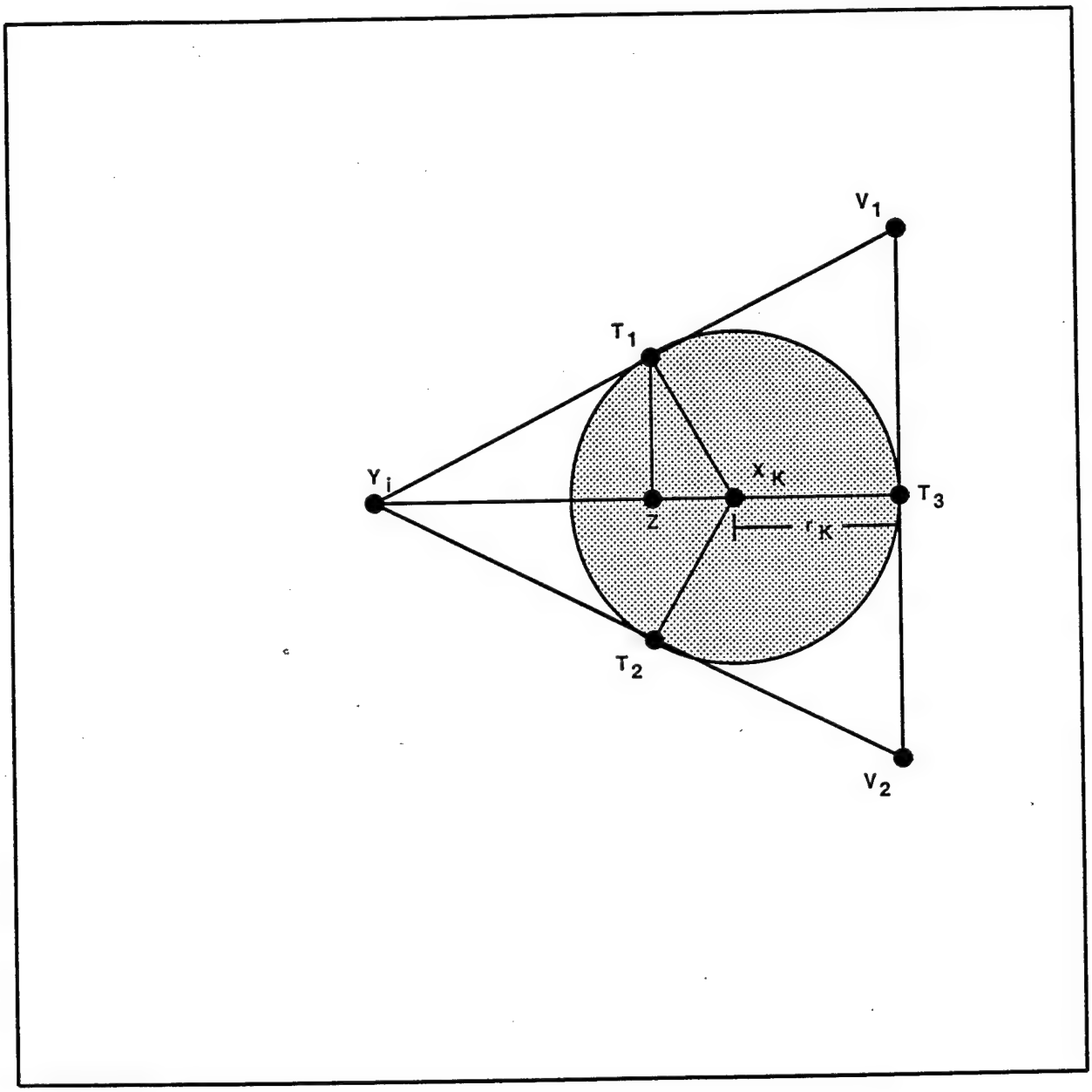


Figure 6: Triangle with Vertex Y_i Circumscribing a Circular Threat Region

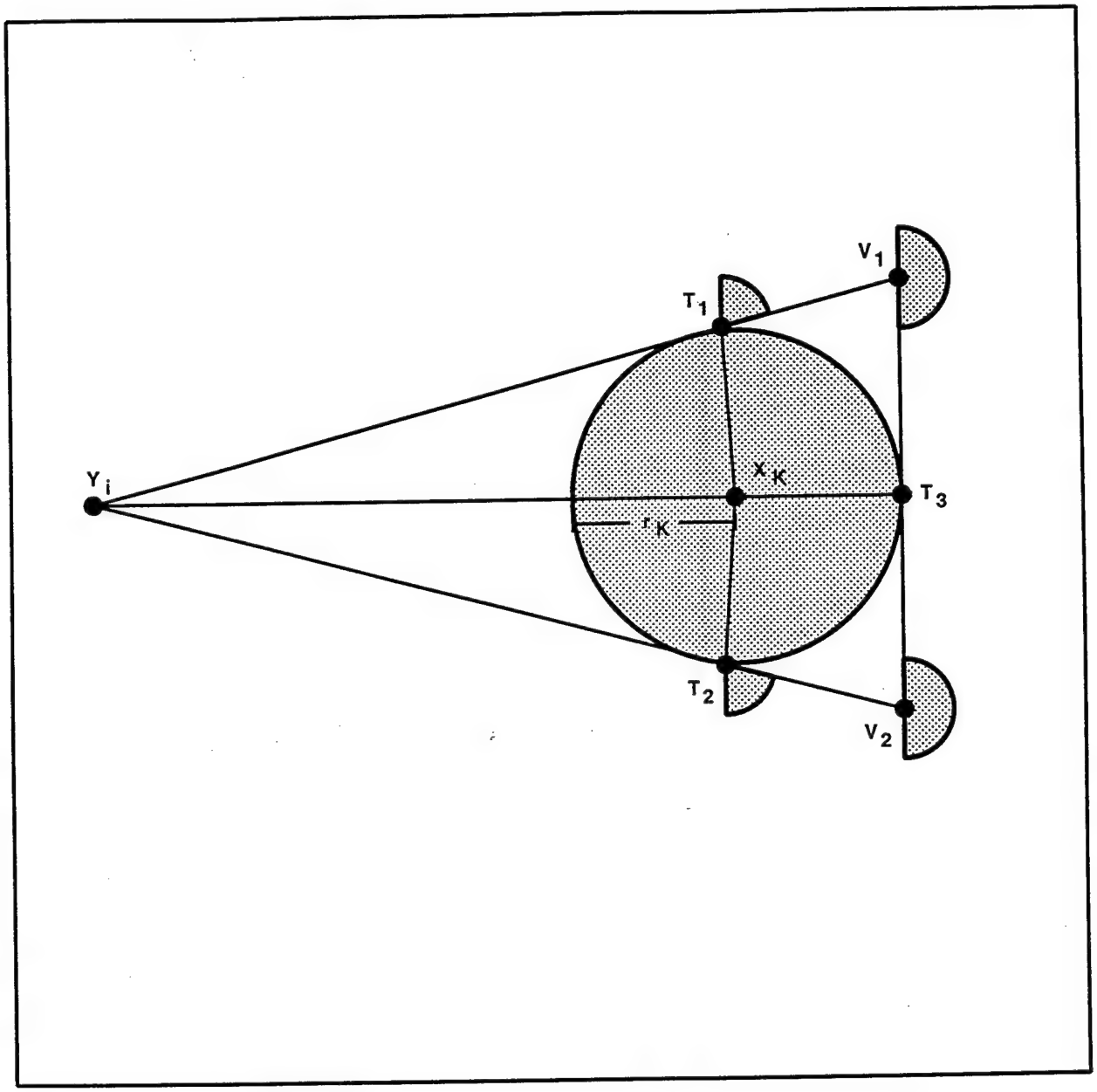


Figure 7: Opposite Side Field of View at Points V_1, V_2, T_1, T_2

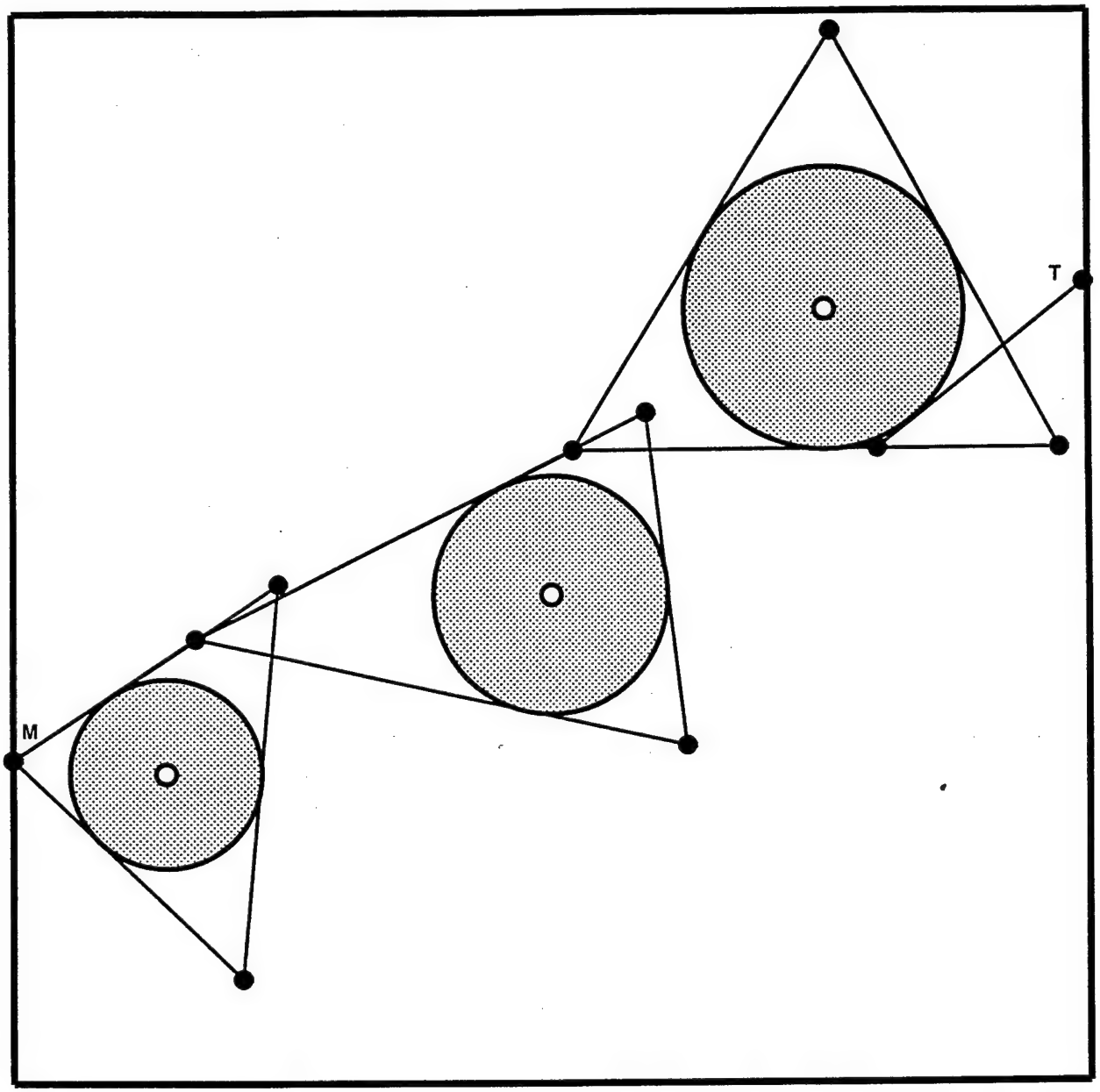


Figure 8: Path Produced by the Circumscribed Triangle Algorithm

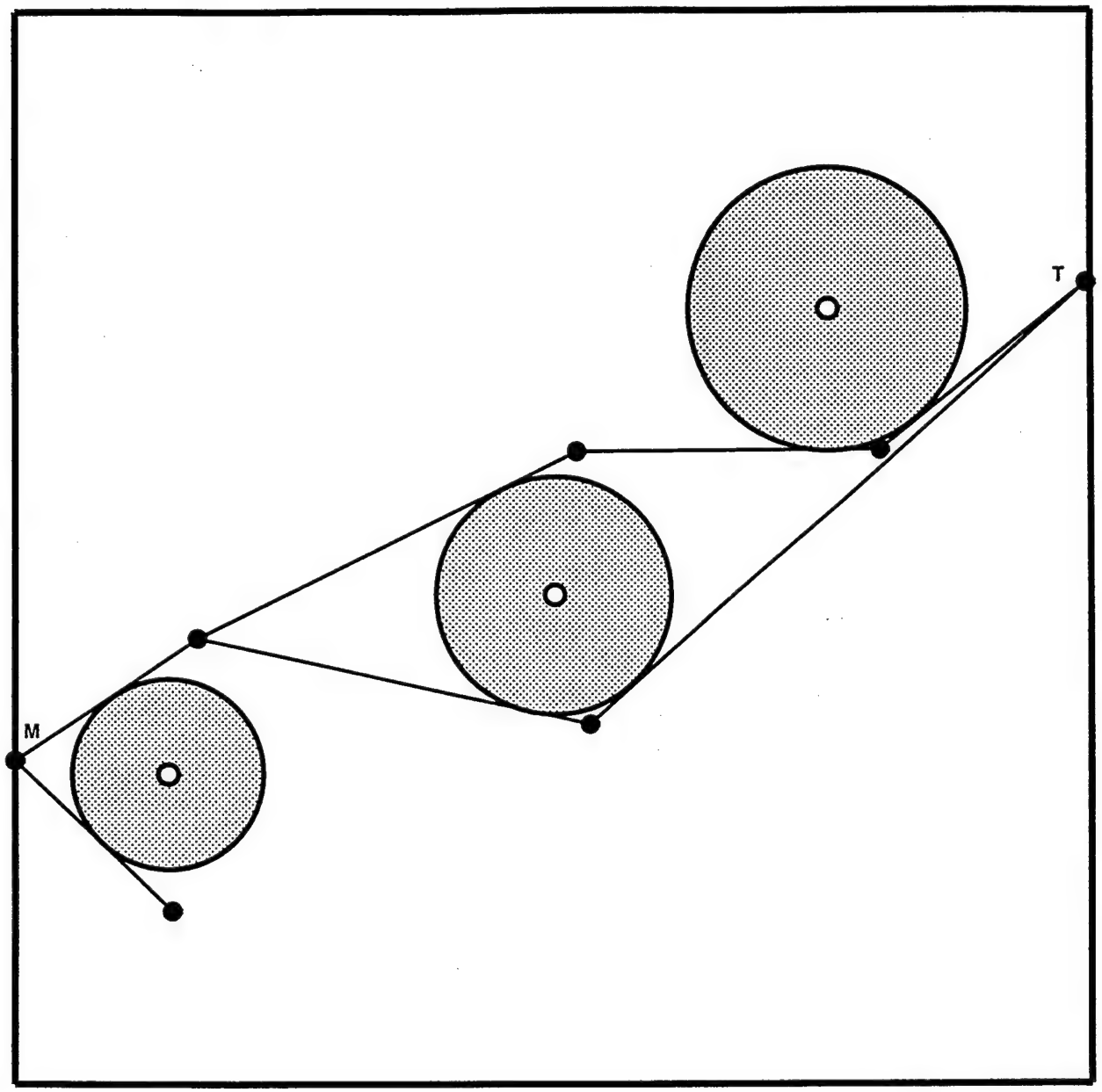


Figure 9: Partial Paths Considered by the Circumscribed Triangle Algorithm

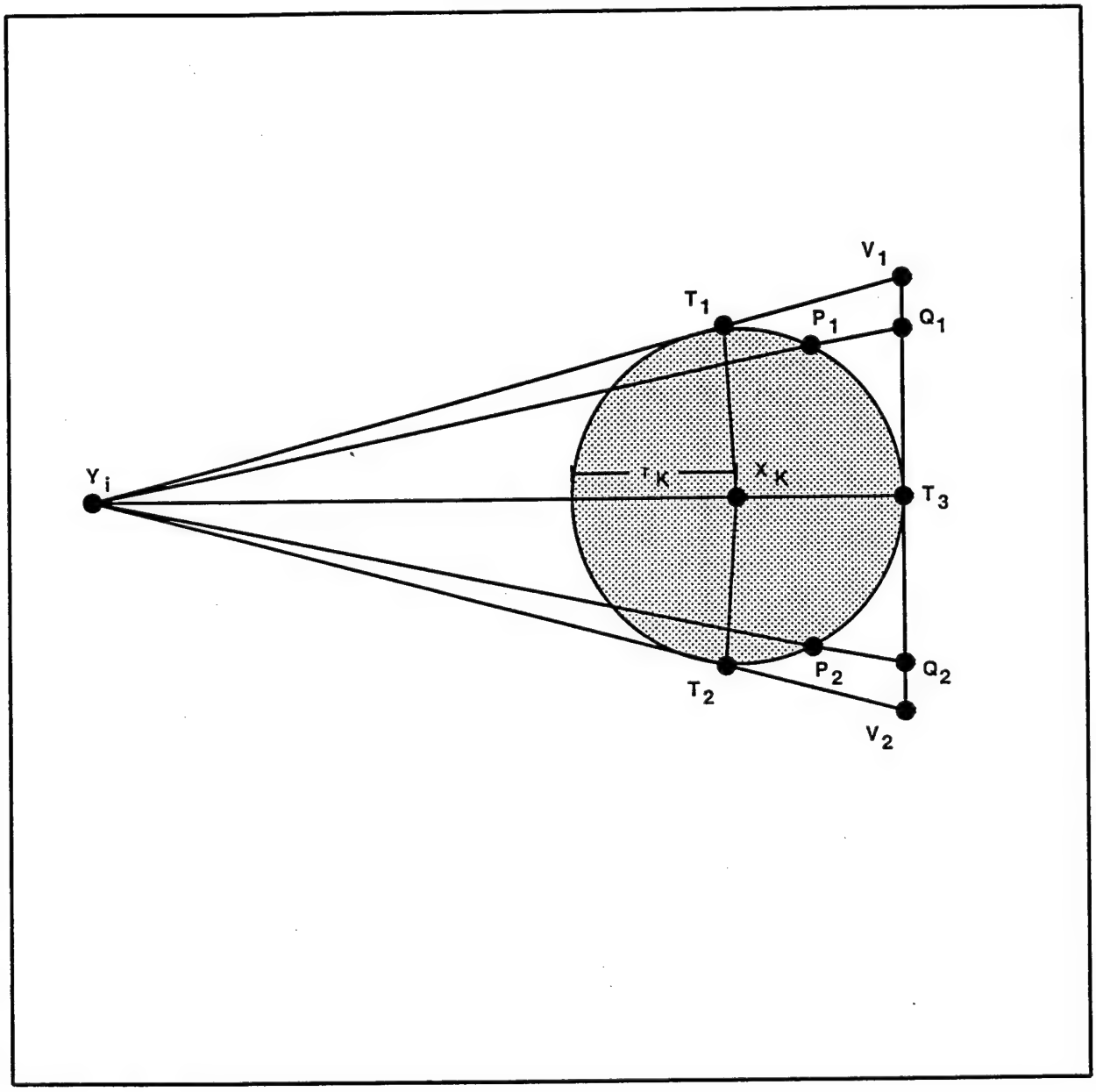


Figure 10: Circumscribing Triangle and Incursion Triangle with Vertex Y_i for Threat Region R_K

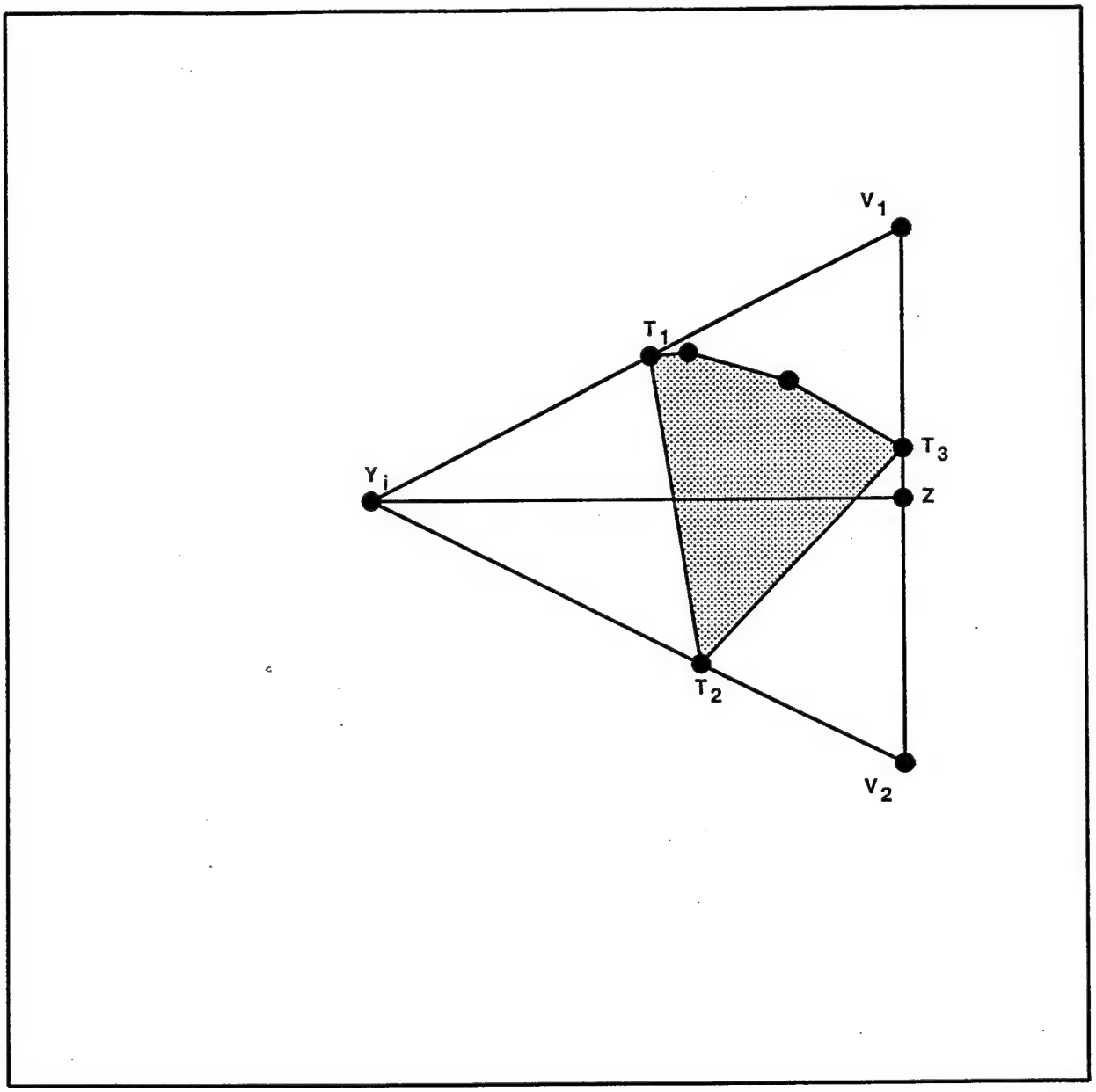


Figure 11: Triangle with Vertex Y_i Circumscribing a Convex Hull Threat Region

Table 1. Comparison of Probabilities for the Mission Planning Problem

Prob	Deterministic Algorithm		Probabilistic Algorithm					
			90%		75%		60%	
	Dist	Segmts	Dist	Segmts	Dist	Segmts	Dist	Segmts
1	957	3	955	3	952	3	949	3
2	853	3	849	3	842	3	834	3
3	853	4	851	3	843	3	803	3
4	945	3	939	3	925	3	905	3
5	795	2	794	2	789	2	783	2
6	878	4	876	4	868	4	857	4
7	1042	3	1040	3	1031	3	1012	3
8	805	2	805	2	804	1	804	1
9	782	2	782	2	780	2	780	1
10	1049	3	1044	3	1031	3	1012	3
11	847	3	845	3	838	3	835	2
12	820	2	819	2	817	2	815	1
13	810	2	809	2	807	2	807	1
14	945	3	942	3	861	3	809	2
15	966	3	962	3	950	3	780	2
16	837	3	837	3	835	3	835	1
17	865	3	857	3	848	3	825	3
18	1009	3	1005	3	994	3	977	3
19	900	4	895	3	882	3	857	3
20	862	2	851	3	837	2	820	3
Totals	17820	57	17757	56	17534	54	17099	47
Scaled	1.0	1.0	0.996	0.982	0.984	0.947	0.960	0.825

APPENDIX A

SOLUTIONS OBTAINED USING THE CIRCUMSCRIBED TRIANGLE ALGORITHM

Using a grid size of 780 x 700, we randomly generated five groups of test problems. Each group had five circular threats with radii randomly generated on the interval [50, 150]. The four problems in each group had identical threat sites, but different missile and target locations with the missiles located on the left boundary and targets located on the right boundary. The missile and target locations were randomly selected and were different for each of the twenty test problems. The input for each problem consists of seven points (the missile location, the target location, five threat centers) and five radii.

A graphical system was developed using Tcl and Tk (see Welch [23]). Tcl stands for Tool Command Language and Tk is a toolkit for window programming. The system allowed us to display the strike mission graphically. The paths obtained by the circumscribed triangle algorithm may be found in Figures A.1 through A.5.

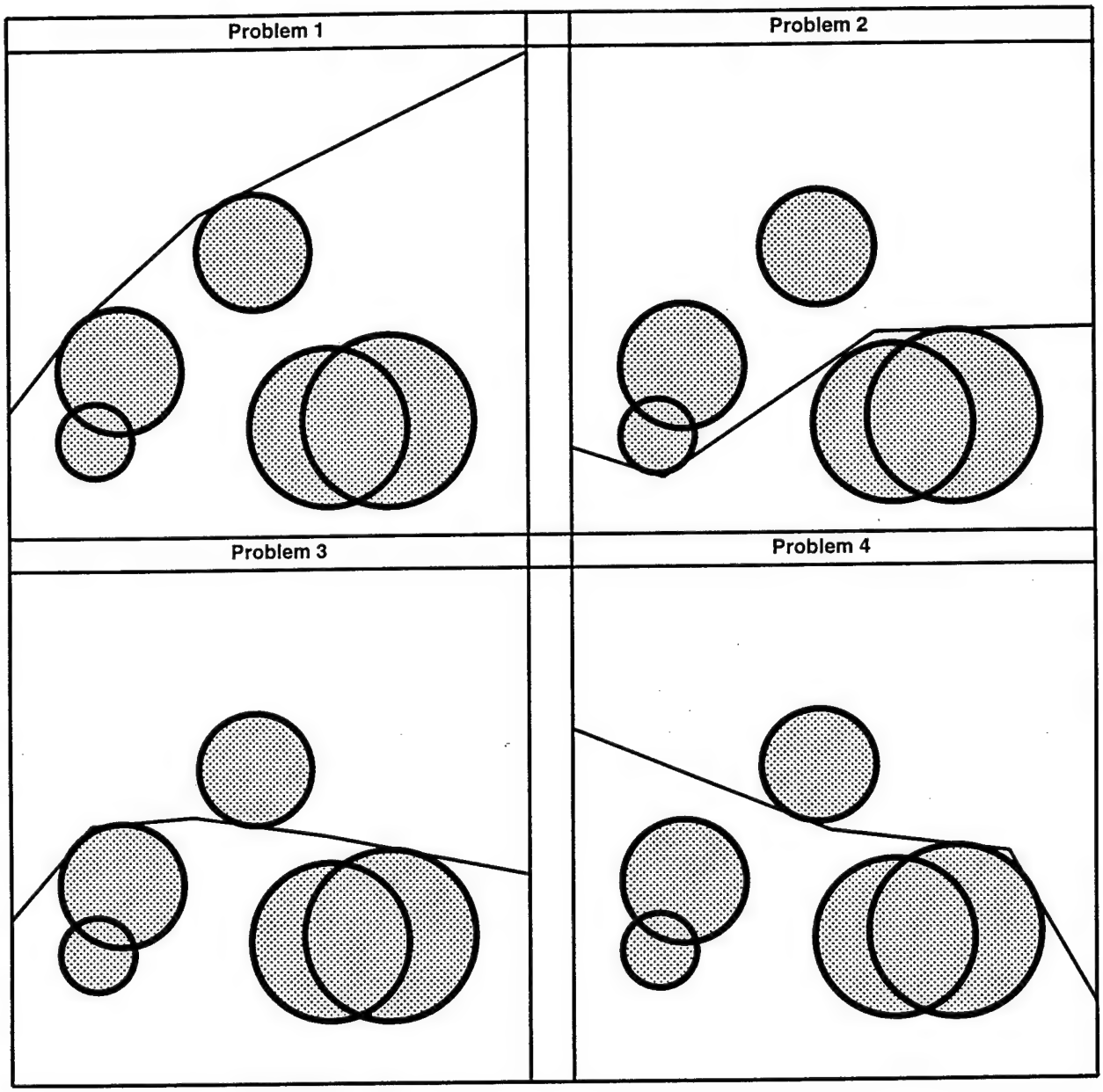


Figure A.1: Display of Problems 1, 2, 3, and 4

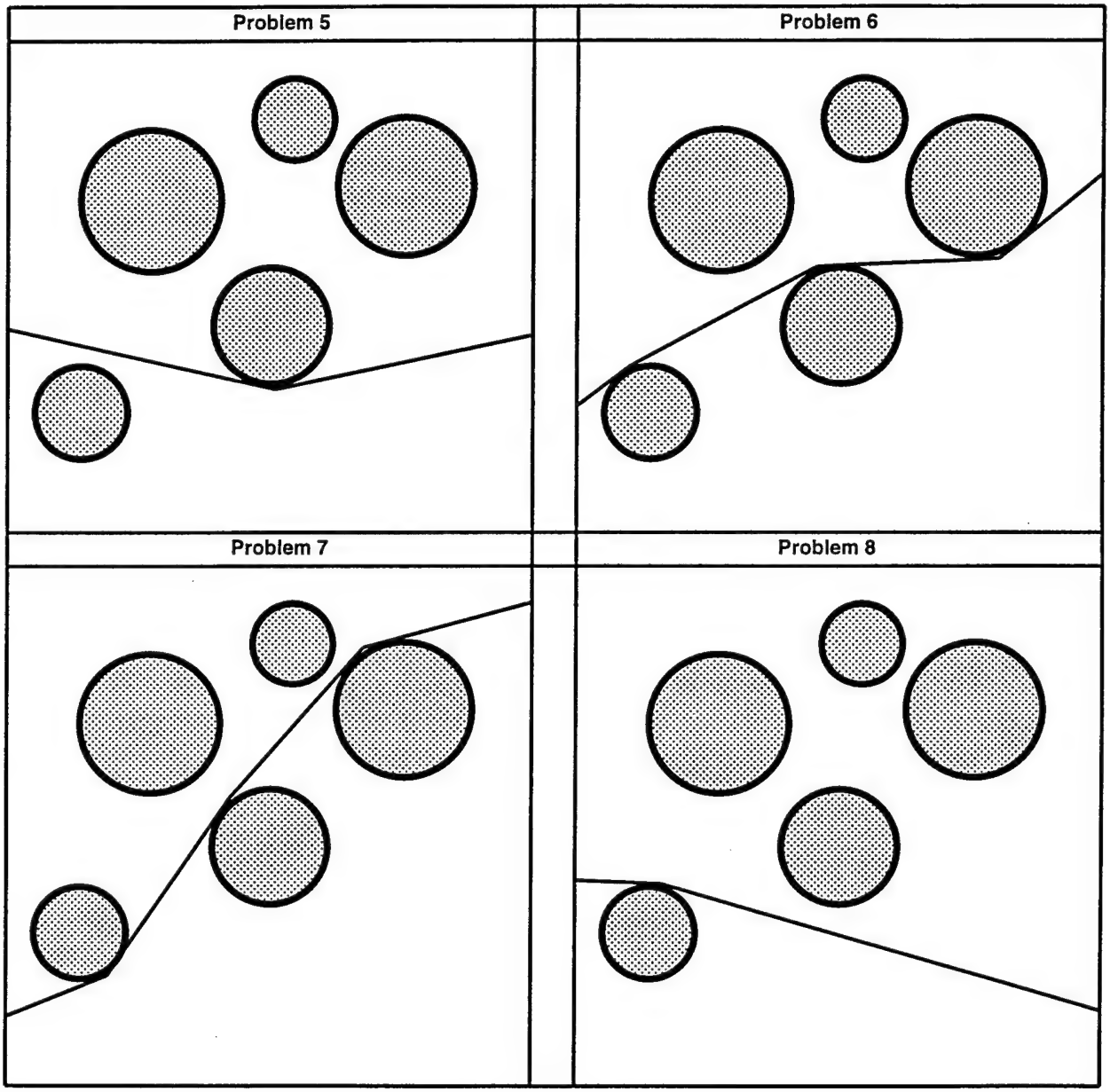


Figure A.2: Display of Problems 5, 6, 7, and 8

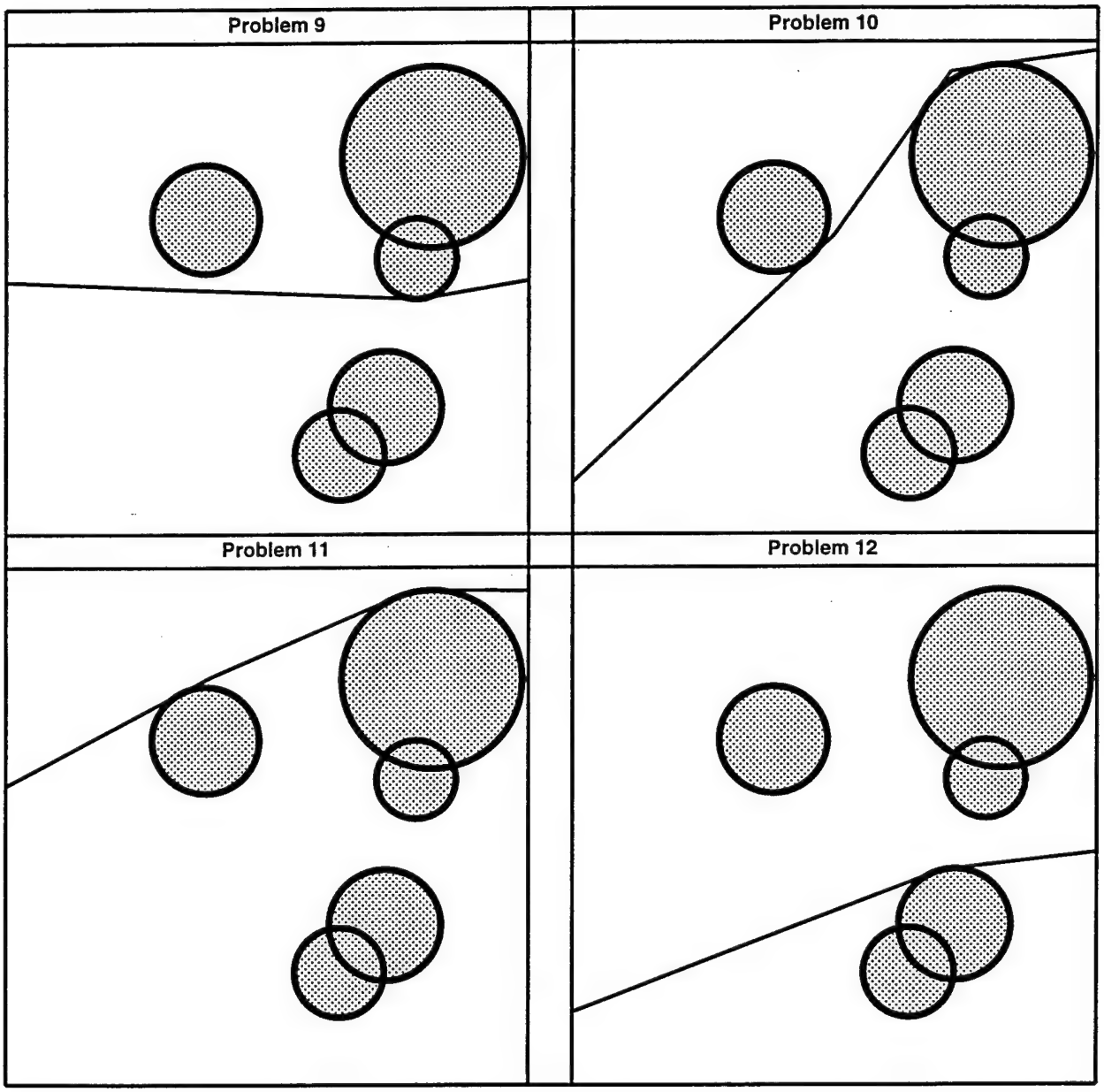


Figure A.3: Display of Problems 9, 10, 11, and 12

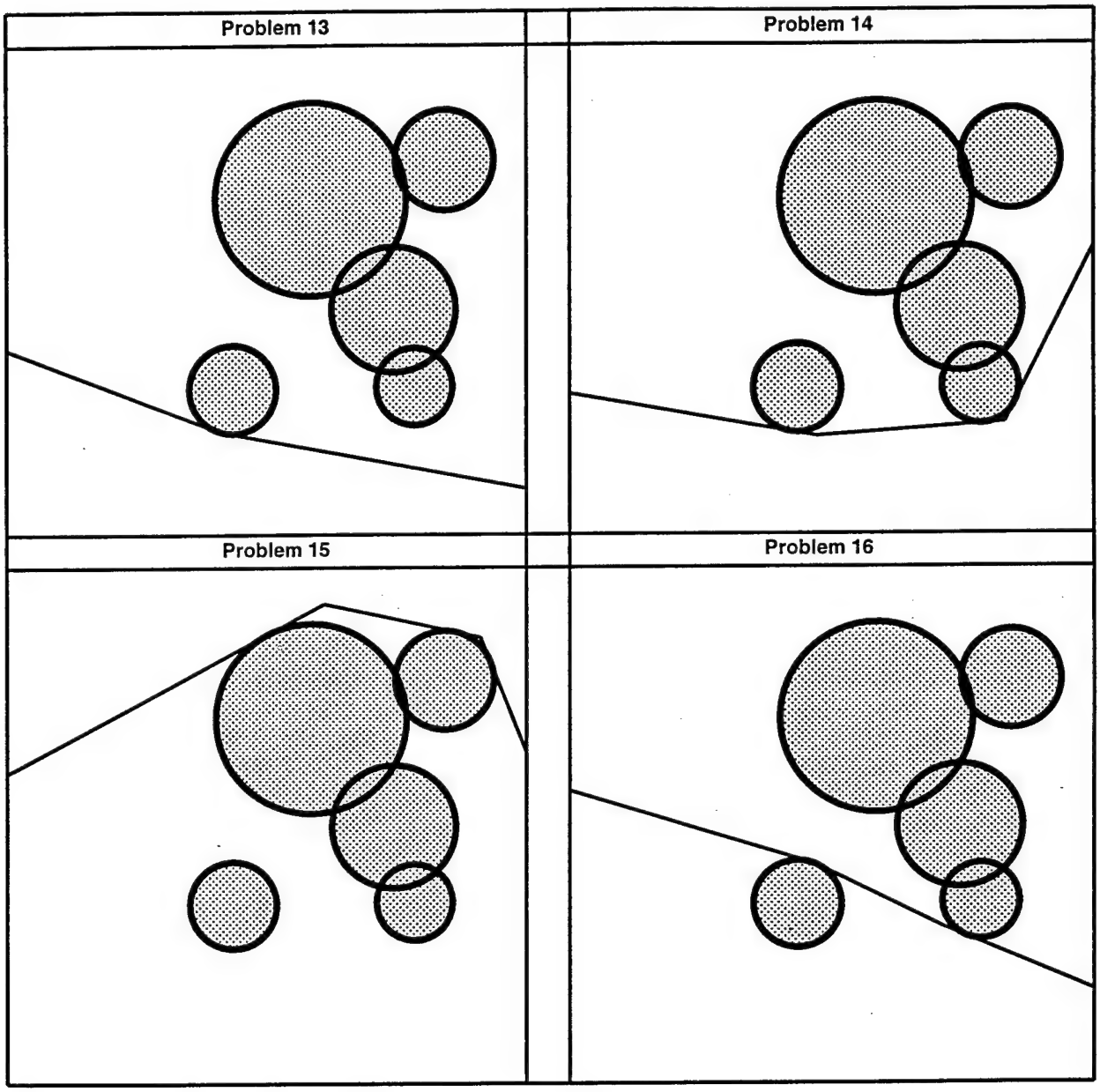


Figure A.4: Display of Problems 13, 14, 15, and 16

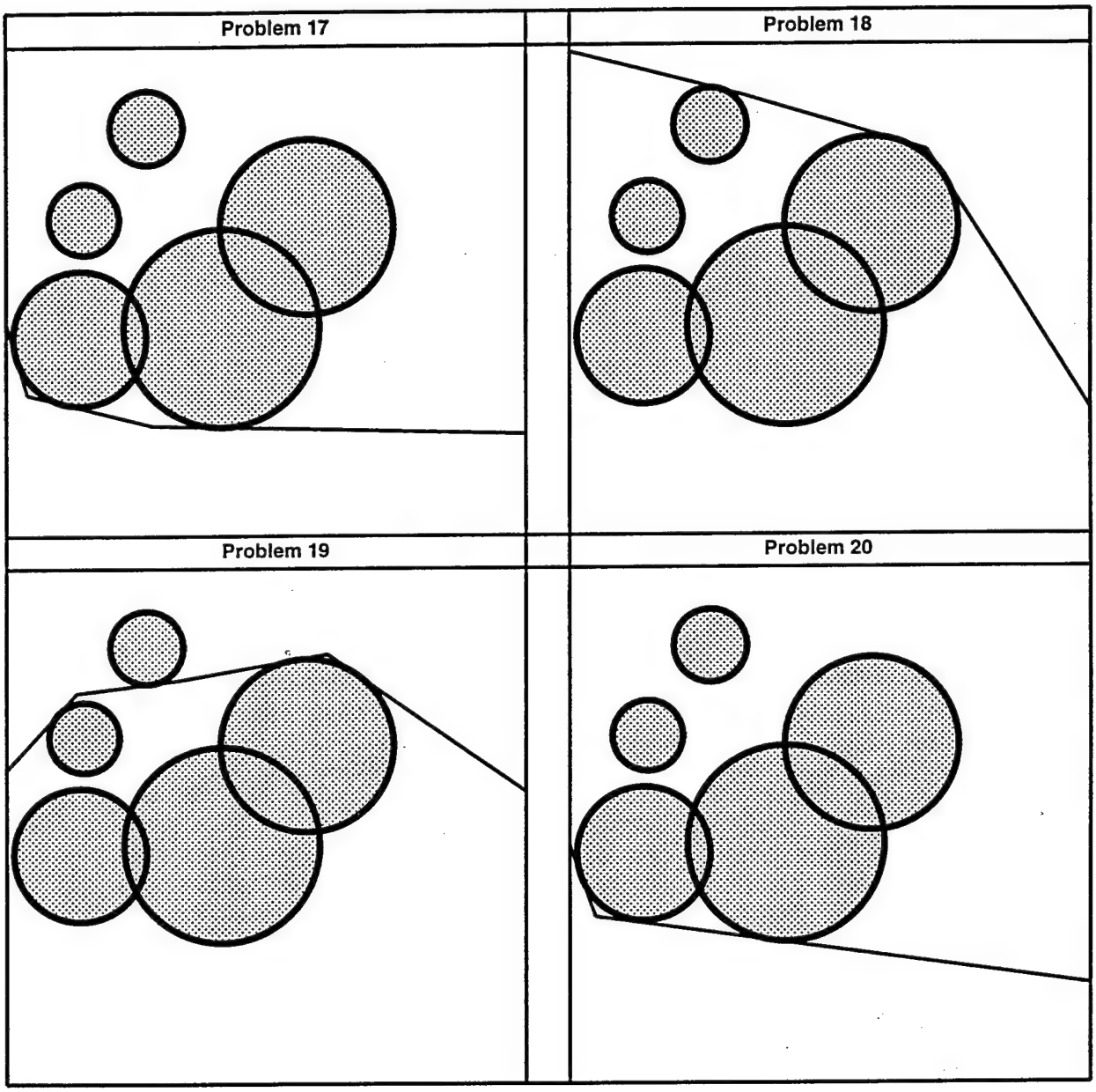


Figure A.5: Display of Problems 17, 18, 19, and 20

APPENDIX B

SOLUTIONS OBTAINED USING THE PROBABILISTIC ALGORITHM

This appendix presents a graphical display of the twenty test problems along with the solutions obtained by the deterministic algorithm and the probabilistic algorithm. The black line is the solution obtained with P set to 60% and the gray line is the mission obtained by the deterministic algorithm. The plots were obtained using Tcl and Tk Solver [23], which is part of our experimental computational package.

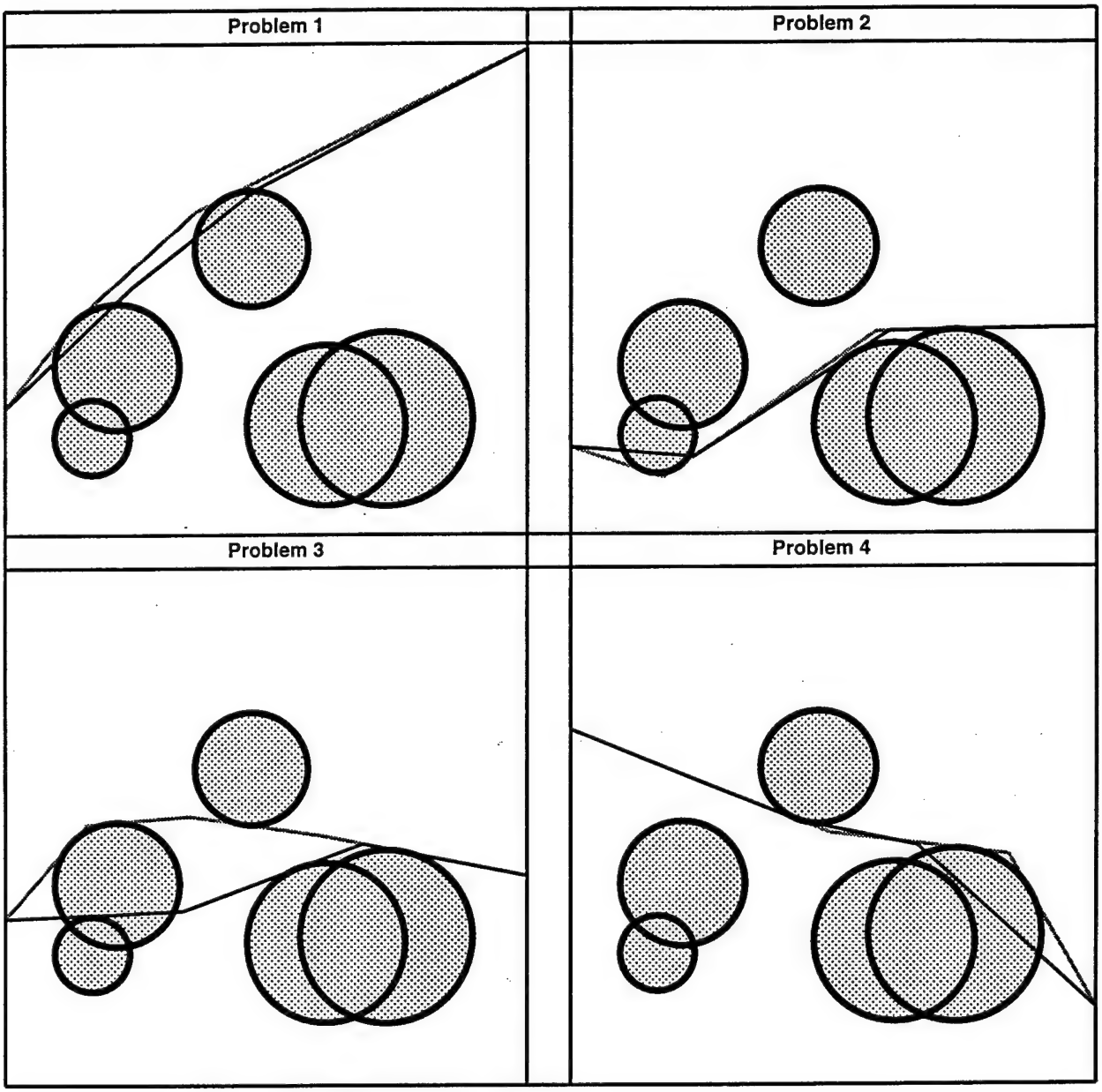


Figure B.1: Display of Problems 1, 2, 3, and 4

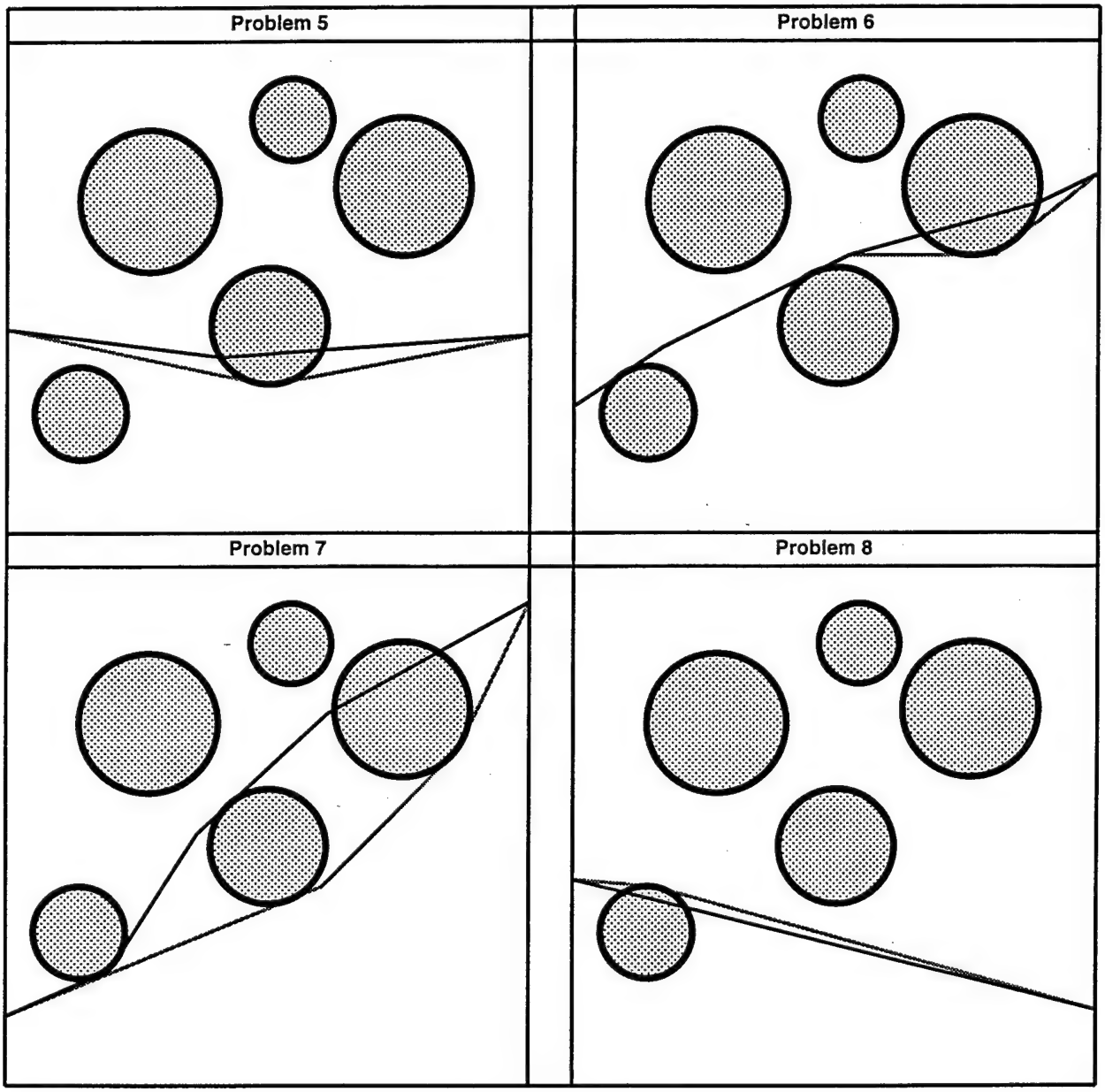


Figure B.2: Display of Problems 5, 6, 7, and 8

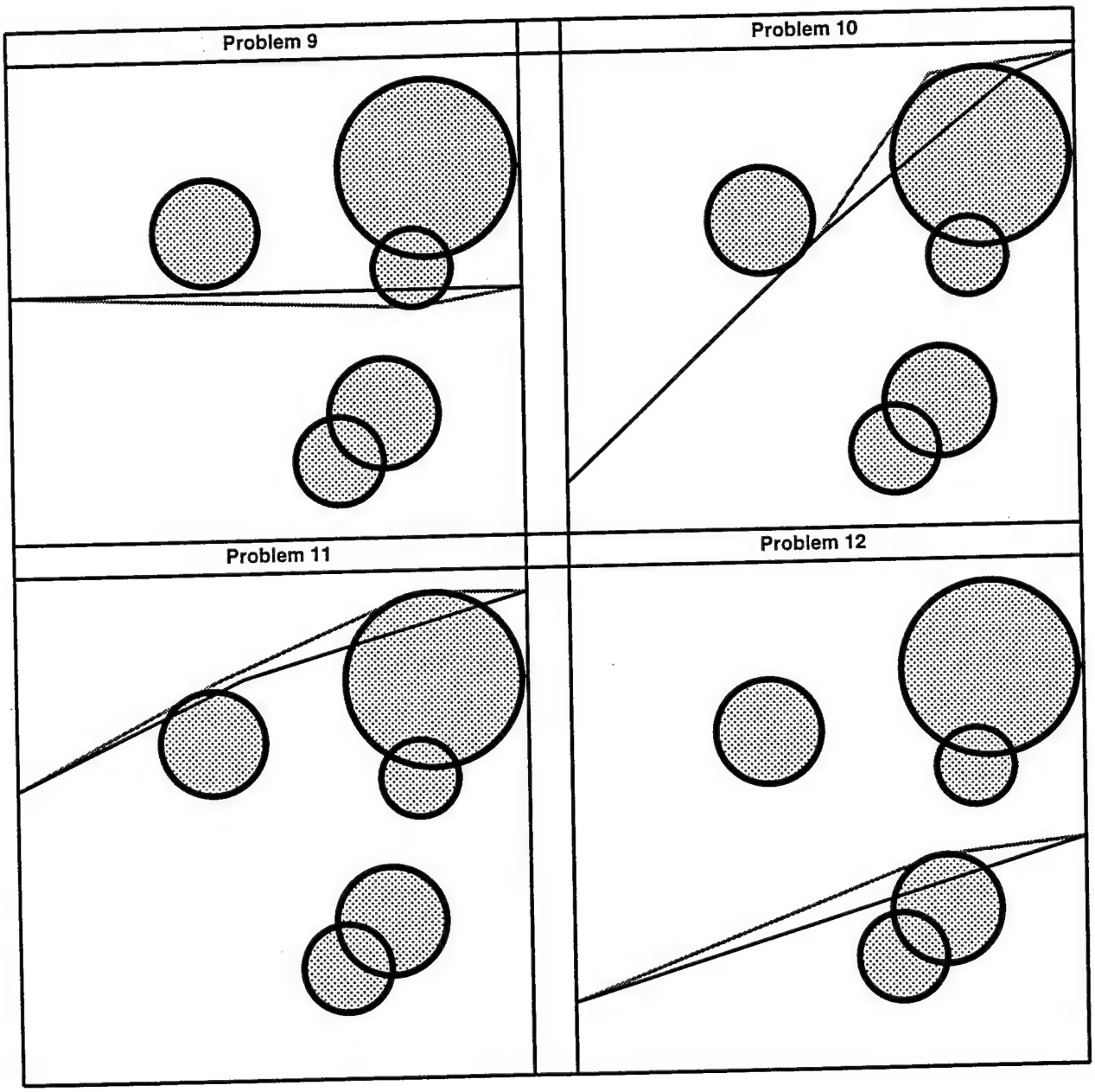


Figure B.3: Display of Problems 9, 10, 11, and 12

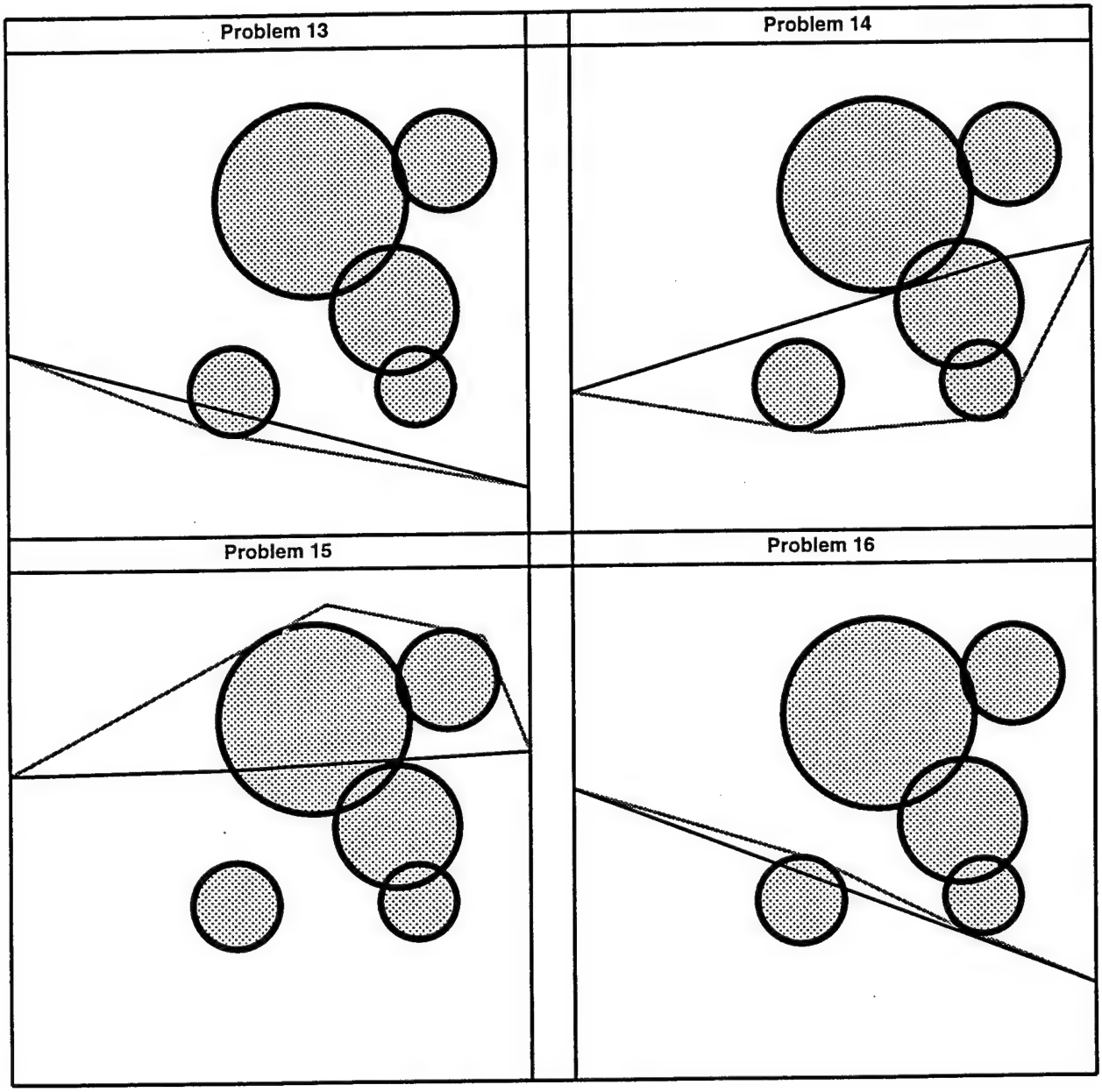


Figure B.4: Display of Problems 13, 14, 15, and 16

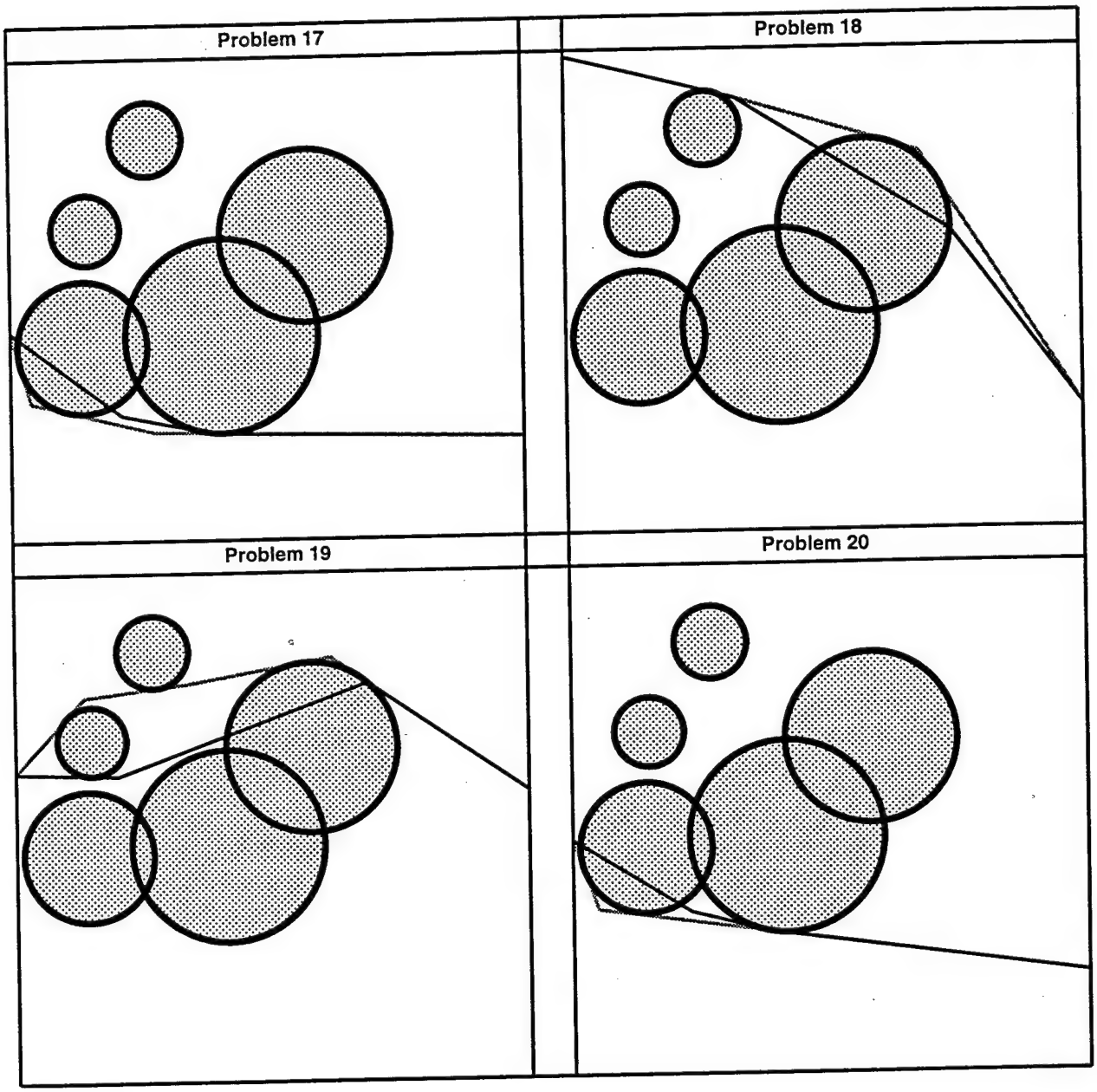


Figure B.5: Display of Problems 17, 18, 19, and 20

APPENDIX C
Algorithm for Threats Modeled as Circles and Polygons

The branch-and-bound algorithm we have developed based on the circumscribed triangle approach is as follows:

Algorithm:

CIRCUMSCRIBE

Inputs:

M	- missle location
T	- target location
K	- number of threats
$R_1, ..., R_K$	- threat regions
Ω	- boundary of the rectangular planning area
Z	- turn angle limit (degrees)

Outputs:

D	- length of best constructed solution path
S	- number of segments in best constructed solution path
$L_1, ..., L_S$	- segments of best constructed solution path

Constructs:

B	- pointer to last point in best constructed solution path
Q	- branch-and-bound queue
$d(E, F)$	- distance from E to F
$L(E, F)$	- line segment from E to F
$H(E, F)$	- halfline from E through F
R_i'	- interior of threat region R_i
I	- number of points constructed
Y_i	- i^{th} constructed point
B_i	- pointer to point previous to Y_i
D_i	- path distance prior to Y_i
S_i	- path segments prior to Y_i
E_i	- set of threat regions eligible for moving away from point Y_i
T_1, T_2, T_3	- constructed tangent points of circumscribled triangle
V_1, V_2	- constructed vertices of circumscribled triangle
V, W	- vertices
β_k	- fall back fraction $(0 < \beta_k \leq 1)$ for segment $T_k V_k$

```
begin
  D ← ∞;  S ← 0;  B ← 0;
  if L(M,T) ∩ R'_i) = Φ  for  i = 1..K then
    D ← d(M,T);  S ← 1;  L_1 ← [M,T];
  else
  begin
    I ← 1;  Y_1 ← M;  B_1 ← 0;  D_1 ← 0;  S_1 ← 0;  E_1 ← {R_1,...,R_K};
    put (Y_1,B_1,D_1,S_1,E_1) on Q;
    while Q ≠ Φ do
    begin
      remove some (Y_i,B_i,D_i,S_i,E_i) from Q;
      b ← B_i;
      select some R_j from E_i;  E_i ← E_i \ R_j;
      if E_i ≠ Φ then put (Y_i,B_i,D_i,S_i,E_i) back on Q;
      form the triangle Y_iV_1V_2 circumscribing R_j having
      its tangent points T_1 ∈ Y_iV_1, T_2 ∈ Y_iV_2, T_3 ∈ V_1V_2;
      for k = 1 to 2 do
      begin
        if 180 − Y_bY_iT_k ≤ Z  and
          L(Y_i,T_k) ∩ R'_l) = Φ  for  l = 1..K  and
          D_i + d(Y_i,T_k) + d(T_k,T) < D then
        begin
          W ← H(E,F) ∩ Ω
          if L(Y_i,T_k) ∩ R'_l) = Φ  for  l = 1..K then
            V ← W
          else
            V ← V_k
          endif
          search for a point U on L(T_k,V) nearest to T_k satisfying:
            180 − Y_iUT ≤ Z  and
            L(Y_i,U) ∩ R'_l) = Φ  for  l = 1..K  and
            L(U,T) ∩ R'_l) = Φ  for  l = 1..K;
          if such a point U exists then
          begin
            if D_i + d(Y_i,U) + d(U,T) < D then
            begin
              I ← I + 1;
              Y_I ← U;  B_I ← i;  D_I ← D_i + d(Y_i,U);  S_I ← S_i + 1;
              m ← I;  I ← I + 1;
              Y_I ← T;  B_I ← m;  D_I ← D_m + d(U,T);  S_I ← S_i + 2;
              D ← D_I;  S ← S_I;  B ← I;
            end
          else
          begin
            if E_i ≠ Φ then
            begin
              compute β_k such that U ← (1 − β)T_k + βV_k and 180 − Y_iUT_i = Z;
              if L(Y_i,U) ∩ R'_l) = Φ  for  l = 1..K then
              begin
                I ← I + 1;
                Y_I ← U;  B_I ← i;  D_I ← D_i + d(Y_i,U);  S_I ← S_i + 1;  E_I ← E_i;
                put (Y_I,B_I,D_I,S_I,E_I) on Q;
              end
            end
          end
        end
      end
    end
  end
  i ← S;  k ← B;
  while k ≠ 0 do  j ← B_k;  L_i ← [Y_j,Y_k];  i ← i − 1;  k ← j;
end
```

APPENDIX D

SOLUTIONS OBTAINED USING THE EXTENDED CIRCUMSCRIBED TRIANGLE ALGORITHM

Using a grid size of 780 x 700, five groups of test problems were randomly generated. Each group has five threat regions composed of a combination of convex sets and circles. The four problems in each group have the same threat regions, but different missile and target locations. The missile is located on the left boundary and the target is located on the right boundary. The origin and destination for each path were selected at random and are different for each of the twenty test problems.

This appendix presents a graphical display of the twenty test problems along with the solutions obtained by the algorithm. The plots were obtained using Tcl and Tk Solver [23], which is part of our experimental computational package.

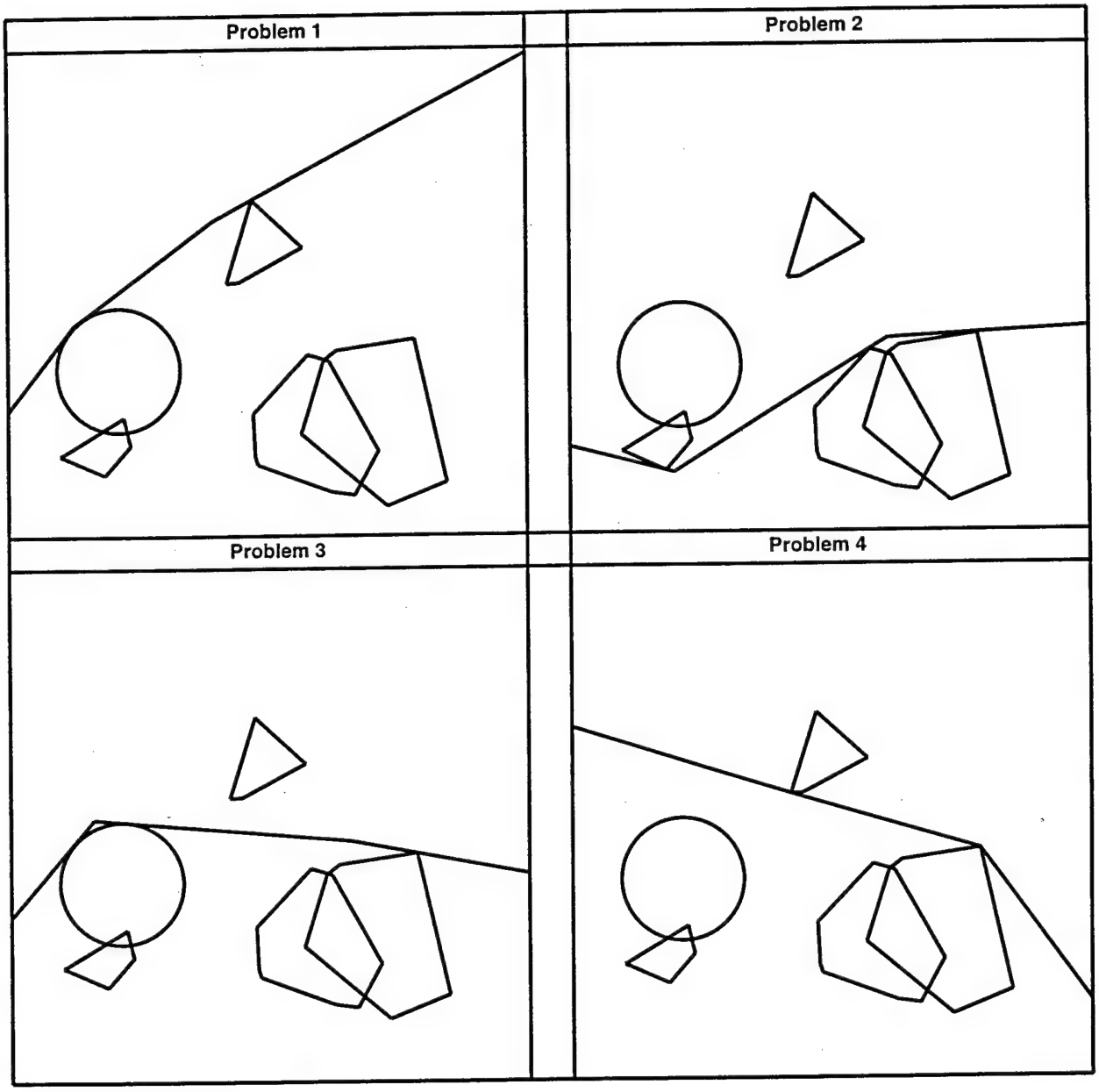


Figure D.1: Display of Problems 1, 2, 3, and 4

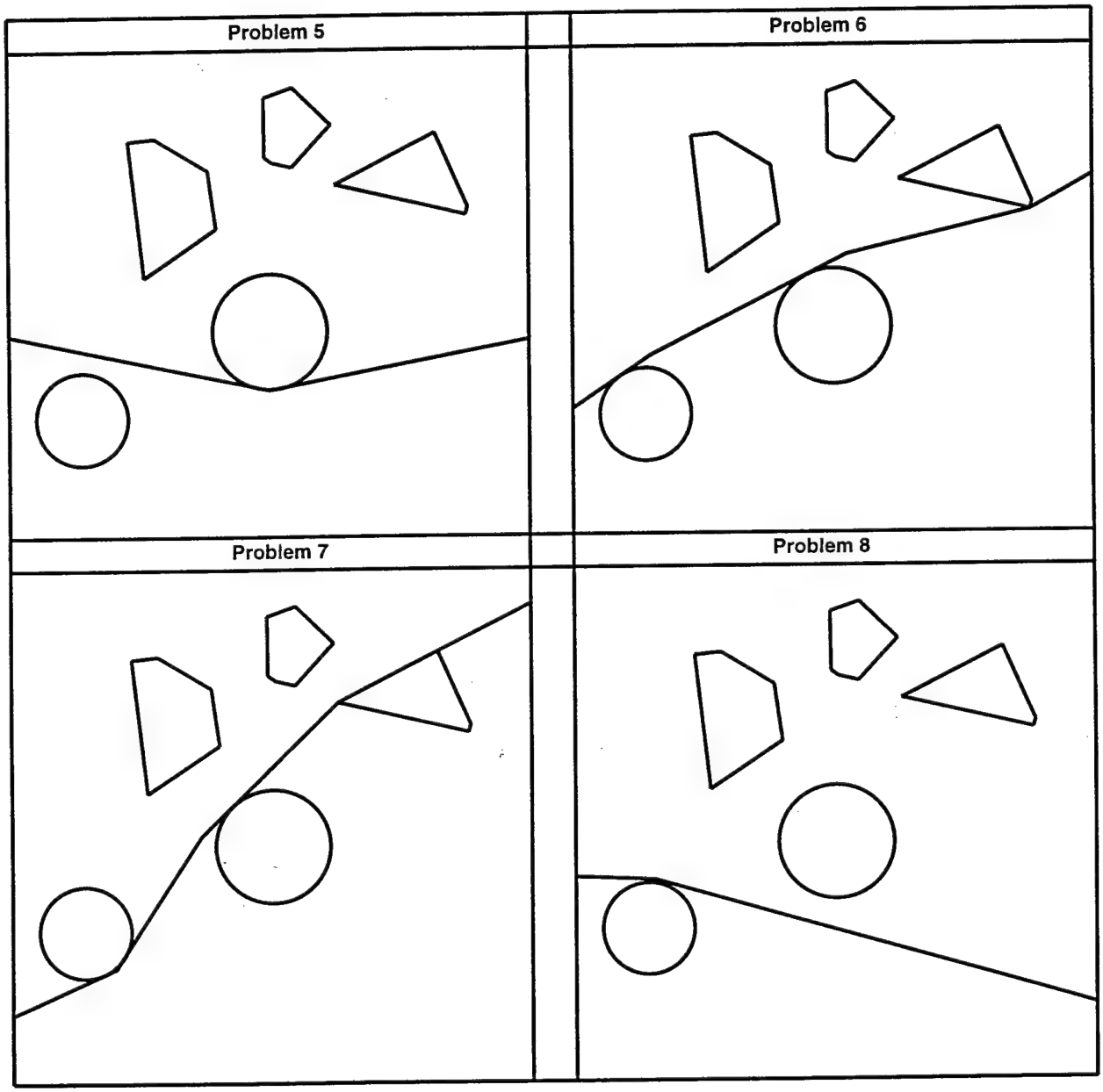


Figure D.2: Display of Problems 5, 6, 7, and 8

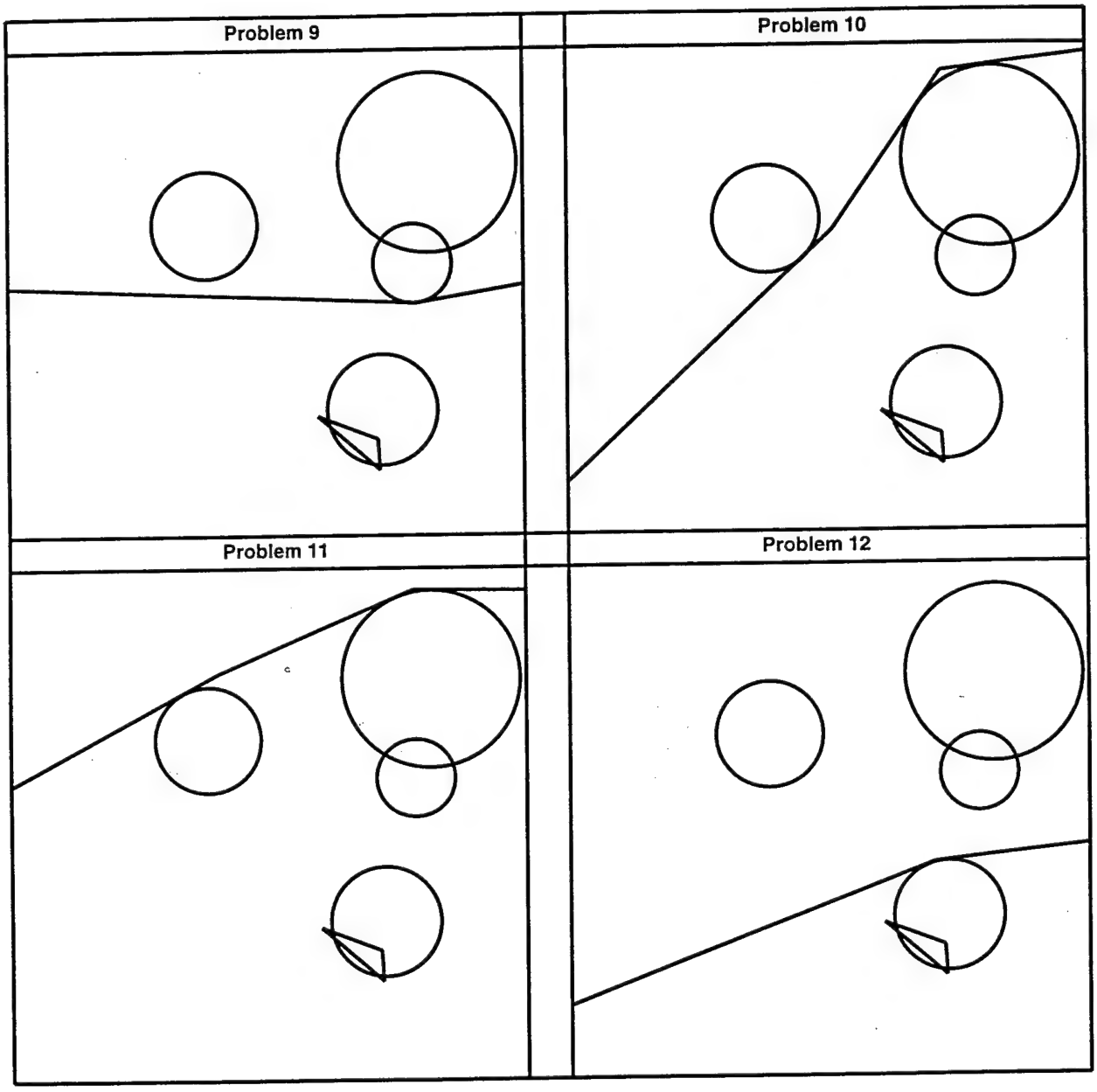


Figure D.3: Display of Problems 9, 10, 11, and 12

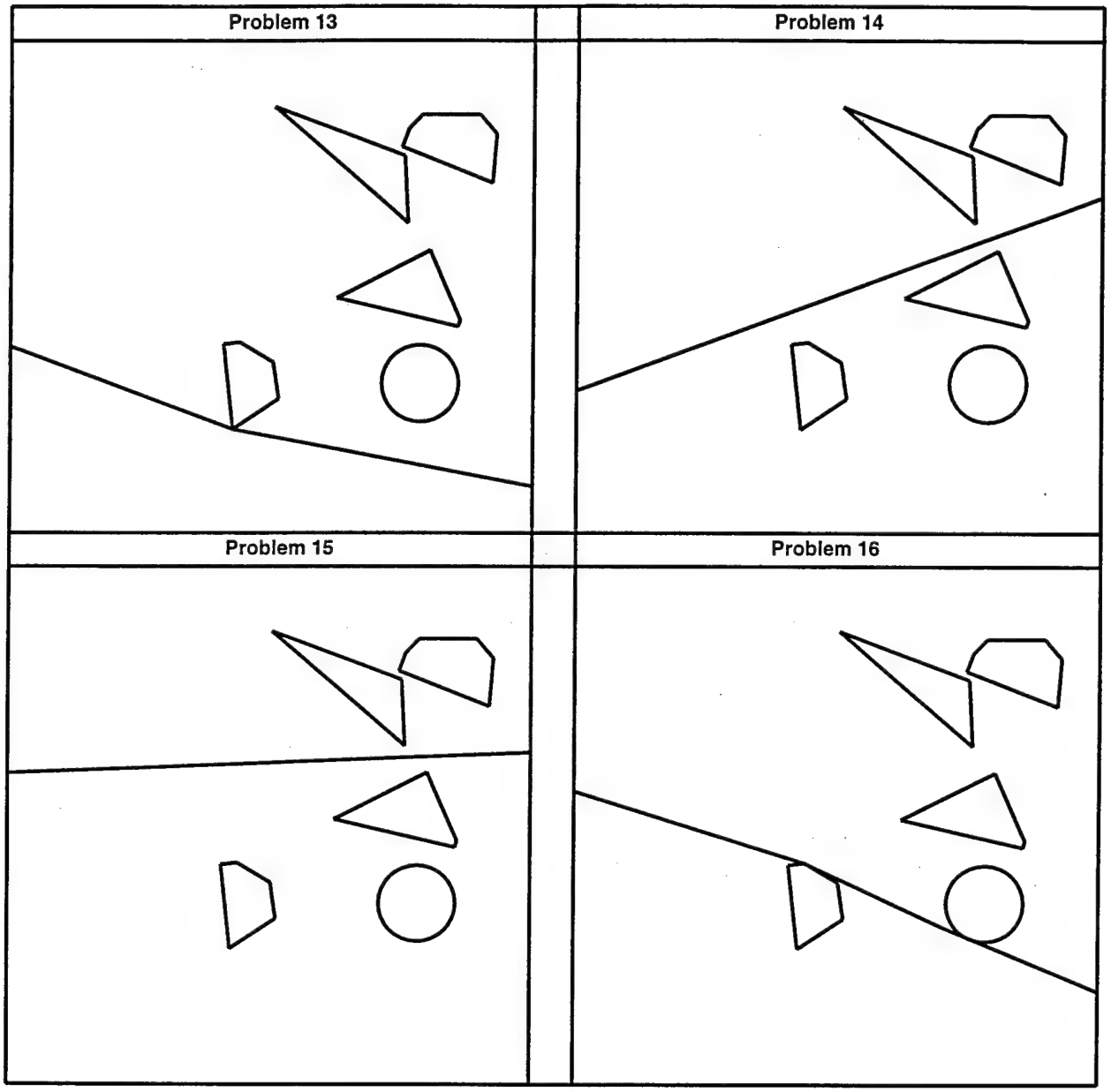


Figure D.4: Display of Problems 13, 14, 15, and 16

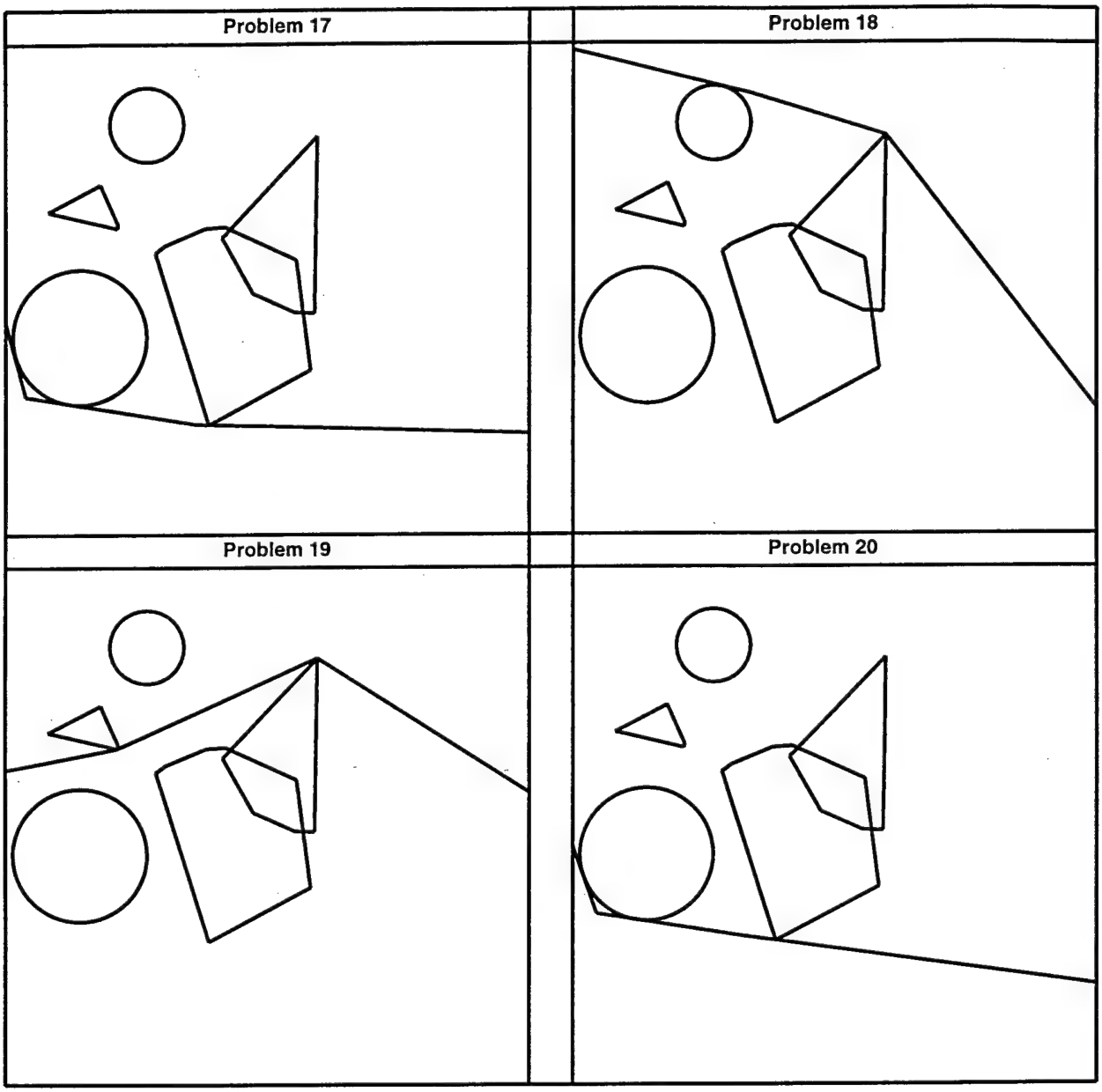


Figure D.5: Display of Problems 17, 18, 19, and 20

Appendix B

Strike Planning Path Generation

Technical Report 97-CSE-24

Cruise Missile Strike Planning: Automatic Multiple Path Generation

by

R. V. Helgason
(helgason@seas.smu.edu)

J. L. Kennington
(jlk@seas.smu.edu)

K. R. Lewis
(klewis@seas.smu.edu)

Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122

November 1997

Abstract

In a military theater, a threat region can be modelled as a convex set in R^2. A mission plan for a cruise missile is a path from its launch location to its target location which avoids the threat regions, if possible, and does not violate a turn angle restriction. A strike plan for a cruise missile strike involving n missiles is a set of n unique mission plans, all of which avoid the threat regions, if possible. This manuscript presents a new algorithm for automatically creating a strike plan in the presence of threats. The new algorithm is based on the two-dimensional geometry of the problem and mimics a strategy which a human operator could easily apply. Twenty test cases have been solved and graphical displays are presented for each test case.

Acknowledgement

This research was supported in part by the Office of Naval Research under contract number N00014-96-1-0315.

1 Introduction

There are a variety of missiles which are classified as cruise missiles. In general, they carry a GPS system, a terrain following system, and a vision system which allows them to use TERCOM maps for precise navigation. A *mission plan* for a cruise missile is defined as a path composed of line segments from a launch site to a target site which does not violate any turn restrictions. The missile is programmed with the waypoints defining the mission plan. The GPS navigation system, in conjunction with a TERCOM map system, guides the missile to its intended target without any outside intervention. Once a cruise missile is launched, its mission cannot be modified.

Since cruise missiles are low flying, they are generally not susceptible to attack by enemy SAM sites. However, they are vulnerable to anti-aircraft artillery (AAA), if a ground crew has their weapons aimed at exactly the location over which the missile will fly. For example, if two missiles fly the same mission with a ten minute time interval between flights, the ground crews along the flight path have an excellent chance of destroying the second missile. The current strategy used to defeat AAA is to vary the missions so that no two missiles fly identical missions. Varying a mission by a few

hundred yards to either the right or left is usually sufficient to defeat AAA.

A mission plan is for a single missile and a collection of mission plans is called a *strike plan.* During recent conflicts in the Persian Gulf, cruise missile strike plans involved approximately 25 missiles. These strikes generally occur early in the morning, involve a two to three hour time window, and require approximately 25 unique mission plans.

In a previous investigation [6], a new algorithm was developed to automatically generate a cruise missile mission plan. The plan (path) is composed of line segments from a given origin (missile location) to a given destination (target location) which avoids threat regions modelled as circles and convex polygons. The algorithm is based solely on the geometry of the problem and attempts to mimic the trial-and-error strategy a human mission planner could apply to this graphical problem. The basic strategy for creating straight line paths around obstacles is to move along the edges of triangles which enclose the obstacles. A heuristic strategy is applied to determine the stopping point along the two sides of the enclosing triangle and branch-and-bound is used to ensure short paths. Unlike optimization problems, the mission planning problem as described in [6] has no mathematical objective function. The objective is to find short feasible missions with only a few segments. The

objective of this study is to present an algorithm to automatically create a strike plan involving 25 unique missile mission plans.

1.1 Description of the Problem

A mission plan for a single missile is a set of connected line segments from an origin to a destination which does not violate a turn angle restriction and does not intersect any threat region, if possible. The threat regions are modelled as circles and convex polygons. A strike plan is a collection of feasible unique mission plans. A feasible strike plan involving two missiles located at M with target T is illustrated in Figure 1. One mission is defined by the line segments (MA, AT) with turn angle $\angle CAT$ and the other is (MB, BT) with turn angle $\angle TBD$. Another strike plan with the same two missiles and targets is illustrated in Figure 2. Note that the two missions are similar but unique. Missions which differ by a few hundred yards to the left or right are generally different enough to defeat AAA. A feasible strike plan having three missiles is illustrated in Figure 3.

Figures 1, 2, 3 about here

Using the ideas and definitions presented in [6], the problem of finding a feasible mission plan may be stated as follows:

> Given two points M and $T \in R^2$, a turn angle restriction Z,
>
> and K convex sets $R_1, \ldots, R_K$, find a set of line segments
>
> $[Y_1, Y_2]$, $[Y_2, Y_3]$, $\ldots$, $[Y_S, Y_{S+1}]$ with $Y_1 = M$ and $Y_{S+1} = T$
>
> such that no line segment intersects the interior of any $R_1, \ldots, R_K$
>
> and no turn angle exceeds Z.

A strike planning problem involving L missiles is to develop L unique feasible mission plans. For a problem having L missiles, there are an infinite number of feasible strike plans, and our algorithm attempts to create a plan having three essential characteristics: the distance for each of the mission plans is short, each mission uses a relatively small number of waypoints, and no turn angle restriction is violated.

1.2 Survey of the Literature

Several investigations for mission planning cover the area with a grid graph and apply a shortest path procedure to find paths through the grid (see [1, 5, 16, 17, 27, 28]). This strategy converts a problem that is fundamentally continuous into one that is discrete. The resulting paths must be adjusted so that there are only a few line segments, and it is difficult to include the turn angle constraint using this strategy. We believe that the strategy implemented in [6], which exploits the geometric properties of the problem, is a superior approach. We know of no investigations which involve multiple missiles in the same region.

The robotics literature on path planning in the presence of obstacles is also related to the problem of planning missions for cruise missiles. Algorithms for generating shortest paths for mobile robots are generally based on finding the shortest path in the tangent graph (or visibility graph) consisting of the tangents between obstacles (see [7, 10, 11, 12, 13]). When applied to mission planning, this approach can result in numerous very short segments and these ideas do not easily adapt to curved surfaces. We know of no investigations involving path planning for multiple mobile robots.

2 The Strike Planning Algorithm

The K convex threat regions are denoted by $R_1, \ldots, R_K$, each with respective center $X_1, \ldots, X_K$. If threat region R_i is a circular region, it has an associated radius r_i. If threat region R_i is defined as a convex hull, it will have an associated set of g_i generator points $\{G_1^i, \ldots, G_{g_i}^i\}$, in which case the center X_i is defined as the average of the generator points so that

$$X_i = \frac{1}{g_i}(G_1^i + \cdots G_{g_i}^i)$$

The L missiles involved in the strike are assumed to be spread out among P origin-destination pairs of missile locations denoted by $\mu_1, \ldots, \mu_P$ and target locations denoted by $\tau_1, \ldots, \tau_P$, with $\lambda_1, \ldots, \lambda_P$ missiles assigned to the respective origin-destination pairs. Thus λ_i missiles will be fired from missile location μ_i at target location τ_i. Furthermore $L = \lambda_1 + \cdots + \lambda_P$.

If a clear path (not intersecting any threat region) is available from missile location μ_i to target location τ_i, the strike planning algorithm generates λ_i two-segment paths $\{Y_1Y_2^1Y_3, \ldots, Y_1Y_2^{\lambda_i}Y_3\}$, where $Y_1 = \mu_i$, $Y_3 = \tau_i$, and the points $Y_2^1, \ldots, Y_2^{\lambda_i}$ are spread out equally spaced about the midpoint $\frac{1}{2}(Y_1 + Y_3)$ along the line perpendicular to line seqment Y_1Y_3 and passing

through that midpoint. The topmost and bottommost sets of paths in Figure 7 were generated in this manner.

If a clear path is not available from missile location μ_i to target location τ_i, the strike plannning algorithm makes use of the circumscribed triangle algorithm to generate a set of λ_i paths from missile location μ_i to target location τ_i, which will have good separation.

The circumscribed triangle algorithm starts at M (a missile location) and constructs line segments by moving from a given segment point tangentially to the circumference of nearby threat regions, until a suitable straight-line path from a segment point to T (a target location) is found. This basic strategy is embedded in a branch-and-bound framework, so that a minimal distance path of this restricted type not violating turn angle limits will be obtained.

When the strike planning algorithm makes use of the circumscribed triangle algorithm with a given origin-destination pair (μ_i, τ_i), it generates a sequence of λ_i single missile paths with $M = \mu_i$ and $T = \tau_i$. After each single missile path is generated, all threat regions which are tangent to any of the path seqments are expanded slightly. For a circular region, the radius is

slightly increased. For a convex hull region, all generator points $G_1^i, \ldots, G_{g_i}^i$ are projected radially slightly away from the center X_i. This expansion insures that on the next single missile path generation, the same path will not be followed. In many cases, the next path generated will have the same general shape but be deflected slightly away from the previous path at the points of tangency to the threat regions encountered. An example of such a set of paths is the topmost set in Figure 4. In some cases, one or more paths may take a slightly different route. An example of such a set of paths is the topmost set in Figure 23. This indicates that other good paths are close in distance to the minimal length restricted path found. After all λ_i paths have been generated in this manner for a given origin-destination pair (μ_i, τ_i), all modified threat regions are returned to their originally specified shapes.

3 Empirical Analysis

Using a grid size of 780 x 780, five groups of test problems were randomly generated and strike plans were developed. The four problems in each group have the same threat regions composed of a combination of convex polygons and circles, but the origins and destinations of the 25 missiles may be dif-

ferent. All missiles are located on the left boundary with all targets located on the right boundary. Each of the twenty test problems were solved with the strike planning algorithm and the solutions are displayed in Figures 4 through 23. The plots were obtained using Tcl and Tk Solver [23], which is part of our experimental computational package. In all cases, the algorithm produced a strike plan having the desired characteristics.

4 Summary and Conclusions

Using the mission planning algorithm described in [6] as the fundamental computational engine, this manuscript presents a new algorithm to automatically generate a strike plan for a cruise missile strike. A strike having 25 missiles is composed of 25 unique mission plans, each of which consists of a path from its location to its target. Even though two missiles located on the same ship or submarine have the same target, their missions must be different enough so that they are not vulnerable to AAA.

Like the mission planning algorithms presented in [6], this algorithm exploits the geometry of the problem description and mimics the trial-and-error approach currently in use by the Navy's mission planners. Since there is no

single criteria for evaluating a strike plan, the merit of the solutions can only be judged qualitatively by examination of the graphical display of the solution. We believe that the twenty displays presented in this manuscript provide a convincing case that our strike planning algorithm provides good strike plans.

References

[1] A. Boroujerdi, C. Dong, Q. Ma, and B. Moret, "Joint Routing in Networks," undated technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM.

[2] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* 1 (1959) 269 - 271.

[3] M. Galicki, "Optimal Planning of a Collision-Free Trajectory of Redundant Manipulators," *The International Journal of Robotics Research*, 11, 6, (1992) 549-559.

[4] Z. Y. Guo and T. C. Hsia, "Joint Trajectory Generation for Redundant Robots in an Environment with Obstacles," *Journal of Robotic Systems*, 10, 2 (1993) 199-215.

[5] R. V. Helgason, J. L. Kennington, and K. R. Lewis, "Finding Safe Paths in Networks," Technical Report 96-CSE-7, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1996).

[6] R. V. Helgason, J. L. Kennington, and K. R. Lewis, "Cruise Missile Mission Planning: A Geometric Algorithm for Automatic Path Generation," Technical Report 97-CSE-23, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1997).

[7] K. Jiang, L. D. Seneviratne, and S. W. E. Earles, "Three-Dimensional Shortest Path Planning in the Presence of Polyhedral Obstacles," *Proceedings of the Institute of Mechanical Engineers: Journal of Mechanical Engineeing Science, Part C*, 210 (1996) 373-381.

[8] C. V. Jones, "Visualization and Optimization," *ORSA Journal on Computing*, 6, 3 (1994) 221-257.

[9] S. Jun and K. G. Shin, "Shortest Path Planning in Discretized Workspaces Using Dominance Relation," *IEEE Transactions on Robotics and Automation*, 7, 3 (1991) 342-350.

[10] J. K. Lee and S. M. Song, "Path Planning and Gait of Walking Machine in an Obstacle-Strewn Environment," *Journal of Robotic Systems*, 8, 6 (1991) 801-827.

[11] Y. H. Liu and S. Arimoto, "Path Planning Using a Tangent Graph for Mobile Robots Among Polygonal and Curved Obstacles," *The International Journal of Robotics Research*, 11, 4 (1992) 376-382.

[12] Y. H. Liu and S. Arimoto, "Computation of the Tangent Graph of Polygonal Obstacles by Moving-Line Processing," *IEEE Transactions on Robotics and Automation*, 10, 6 (1994) 823-830.

[13] Y. H. Liu and S. Arimoto, "Finding the Shortest Path of a Disk Among Polygonal Obstacles Using a Radius-Independent Graph," *IEEE Transactions on Robotics and Automation*, 11, 5 (1995) 682-691.

[14] E. Lu, J. Ni, and S. M. Wu, "An Algorithm for the Generation of an Optimum CMM Inspection Path," *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control*, 116 (1994) 396-401.

[15] S. K. Singh and M. C. Leu, "Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints," *The International Journal of Robotics Research*, 10, 2 (1991) 171-187.

[16] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Fast Computation of Optimal Paths Using a Parallel Dijkstra Algorithm with Embedded Constraints," *Neurocomputing* 8, (1995) 195 - 212.

[17] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Autorouting Using a Parallel Dijkstra Algorithm with Embedded Constraints," undated technical report, The Naval Surface Warfare Center, Dahlgren, VA.

[18] I. Spangelo and O. Egeland, "Trajectory Planning and Collision Avoidance for Underwater Vehicles Using Optimal Control," *IEEE Journal of Oceanic Engineering*, 19, 4 (1994) 502-511.

[19] S. Sundar and Z. Shiller, "Optimal Obstacle Avoidance Based on the Hamilton-Jacobi-Bellman Equation," *IEEE Transactions on Robotics and Automation*, 13, 2 (1997) 305-310.

[20] F. Trogneux, M. Doquet, P. Mallet, "Taking Environmental Constraints into Account in the Planning of the Extra High Voltage Transmission Network; EDF's Approach," *IEEE Transactions on Power Delivery*, 11, 4 (1996), 1901 - 1906.

[21] C. S. Tseng and T. S. Lue, "Path Planning for Robot Manipulators in Polyhedral Objects Environment," *Journal of Robotic Systems*, 12,10 (1995) 637-646.

[22] D. Wang and Y. Hamam, "Optimal Trajectory Planning of Manipulators with Collision Detection and Avoidance," *The International Journal of Robotics Research*, 11, 5 (1992) 460-468.

[23] B. Welch, *Practical Programming in Tcl and Tk*, Prentice-Hall, Inc., Upper Saddle River, NJ (1995).

[24] T. H. Wu and K. Y. Young, "Path Planning in the Presence of Obstacles Based on Task Requirements," *Journal of Robotic Systems*, 11, 8 (1994) 703-716.

[25] S. Q. Zheng, J. S. Lim, and S. S. Iyengar, "Finding Obstacle-Avoiding Shortest Paths Using Implicit Connection Graphs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15, 1 (1996) 103-110.

[26] Q. Zhu, "Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation," *IEEE Transactions on Robotics and Automation*, 7, 3 (1991) 390-397.

[27] M. Zuniga and P. Gorman, "Threat Site Overflight Modeling for Strike Route Optimization," undated technical report, Naval Research Laboratory, Washington, D.C.

[28] M. Zuniga, J. Uhlmann, and J. Hofmann, "The Interdependent Joint Routing Problem: Description and Algorithmic Approach," undated technical report, Naval Research Laboratory, Washington, D.C.

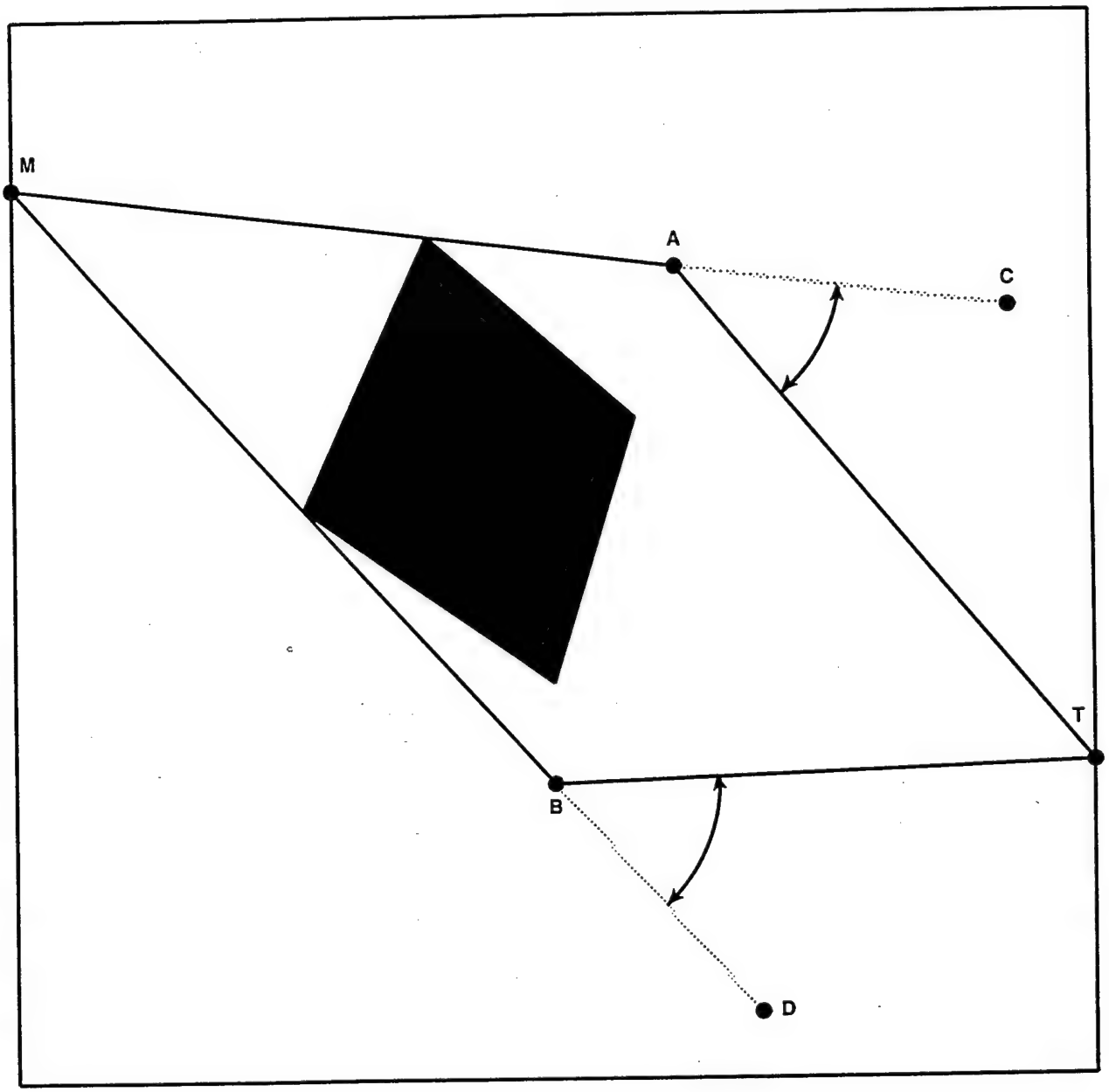


Figure 1: A Strike Plan with Two Missiles

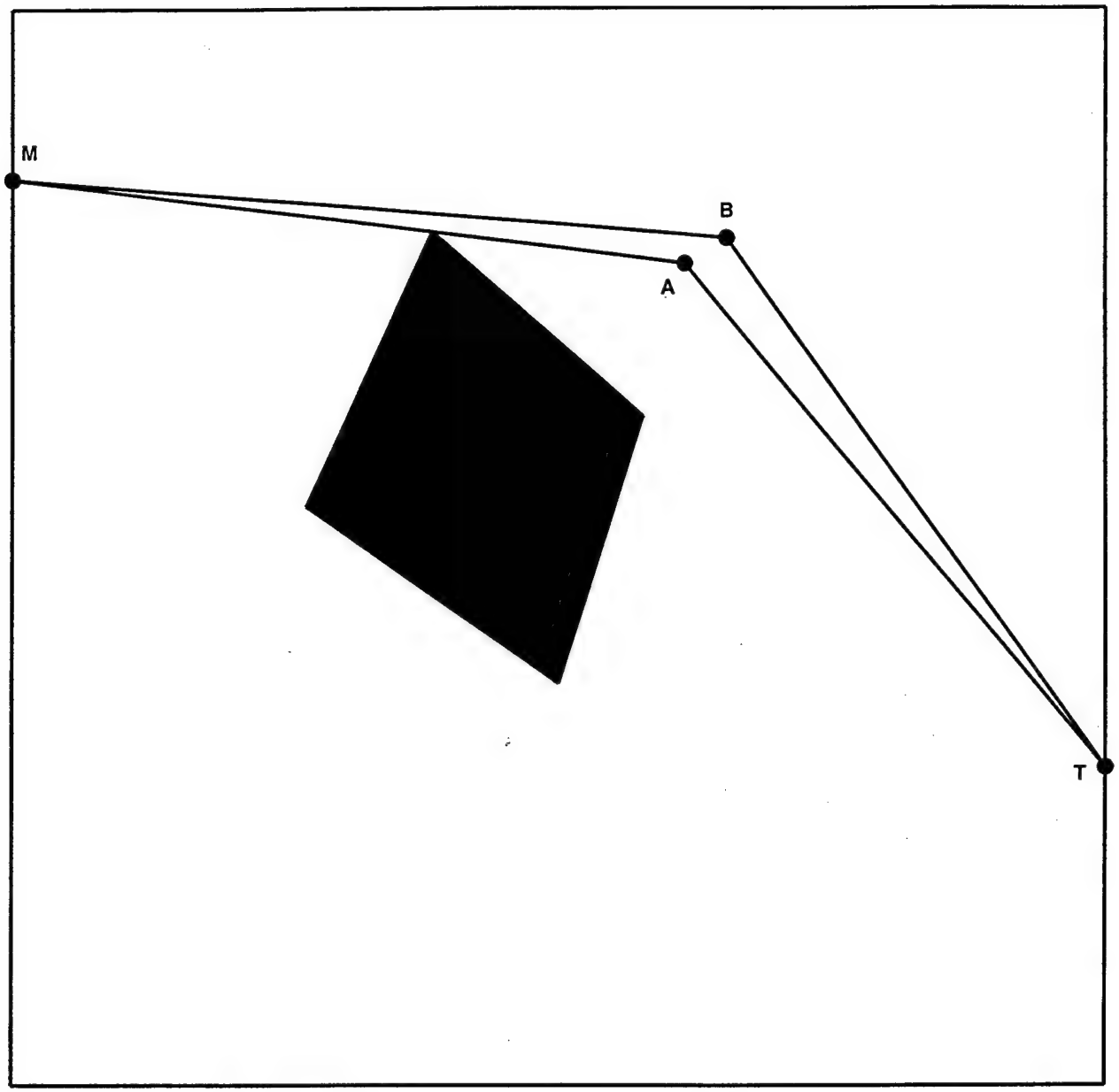


Figure 2: A Two Missile Strike Plan with Similar but Unique Paths

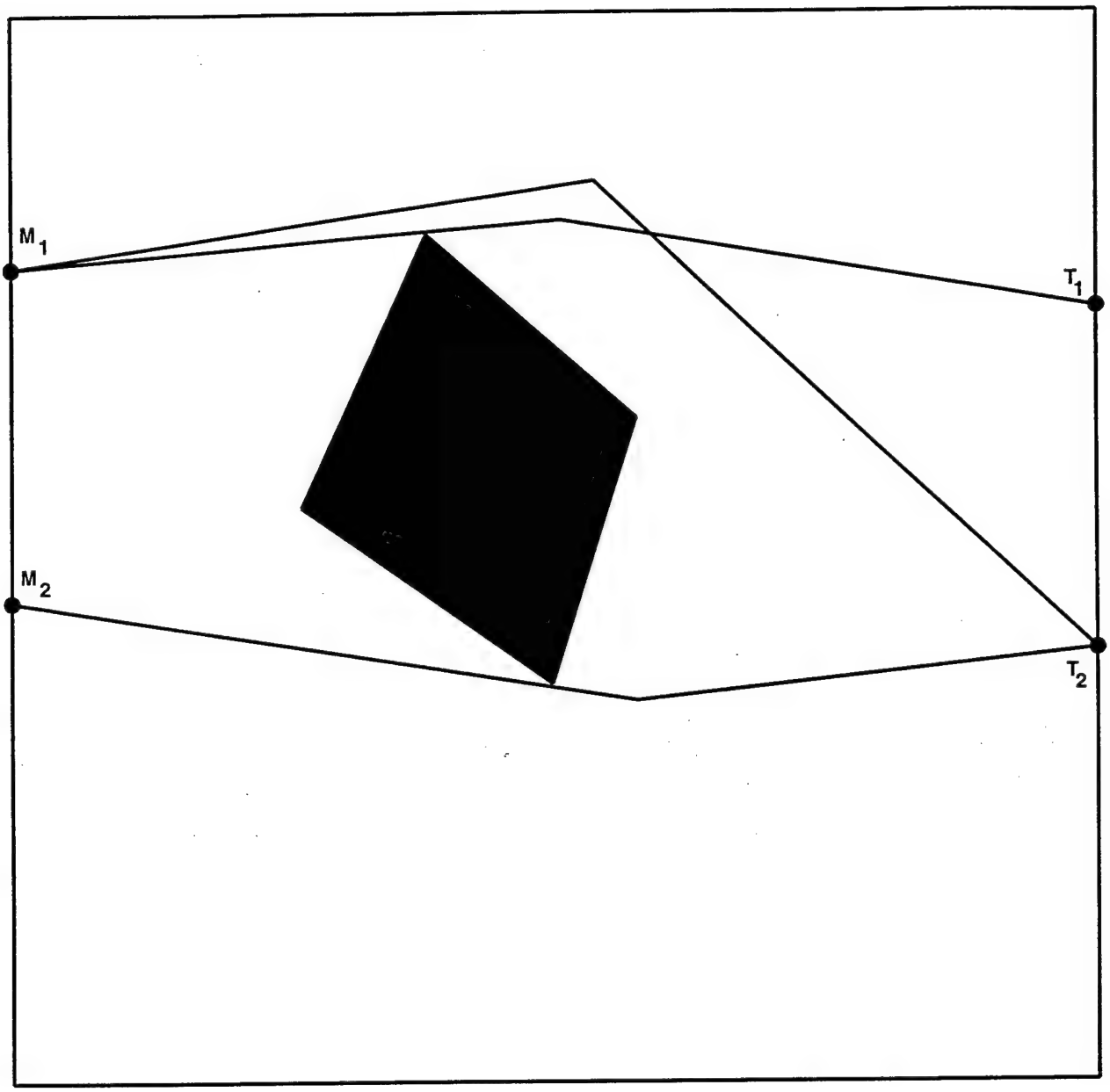


Figure 3: A Three Missile Strike Plan

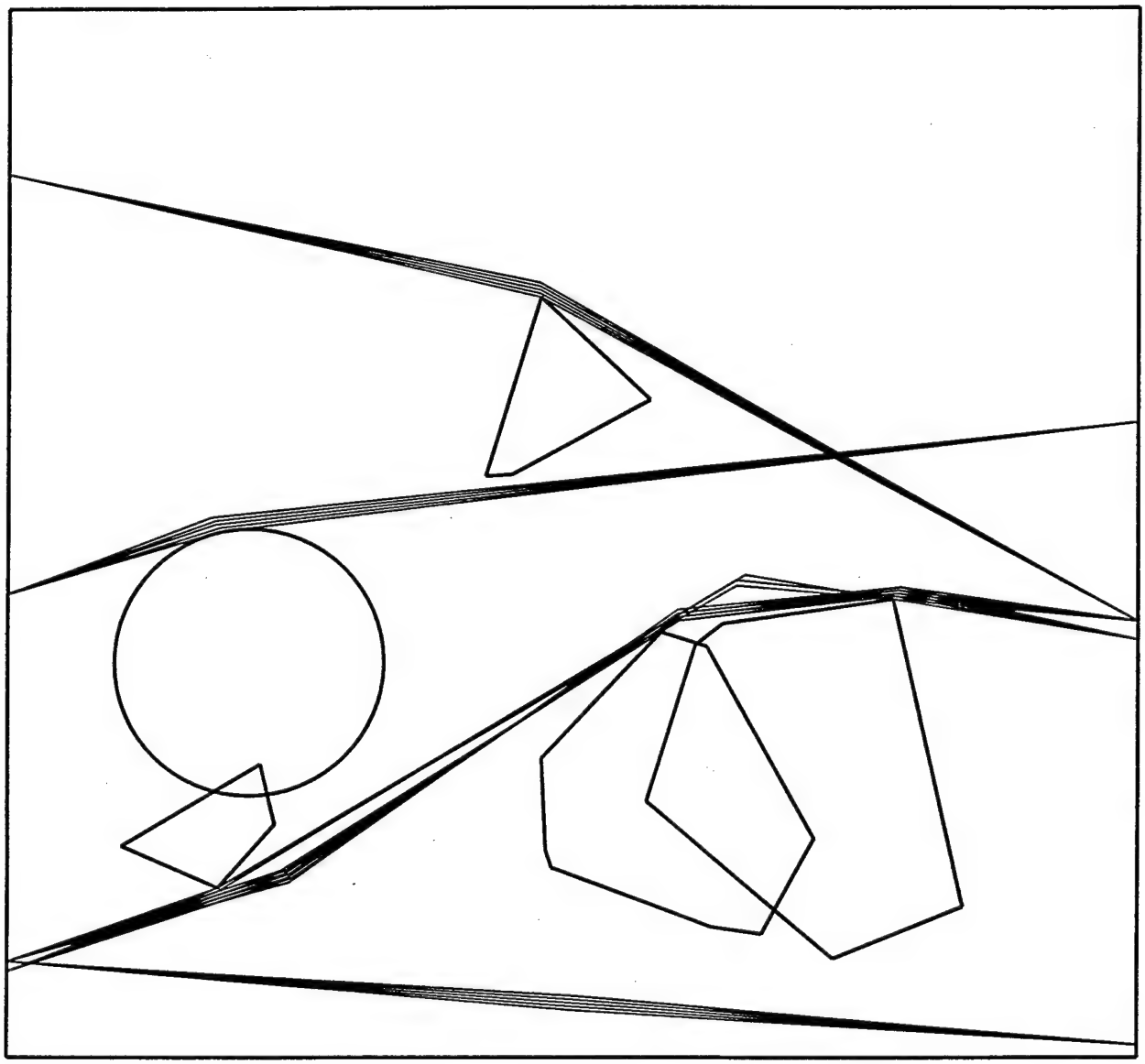


Figure 4: Display of Problem 1

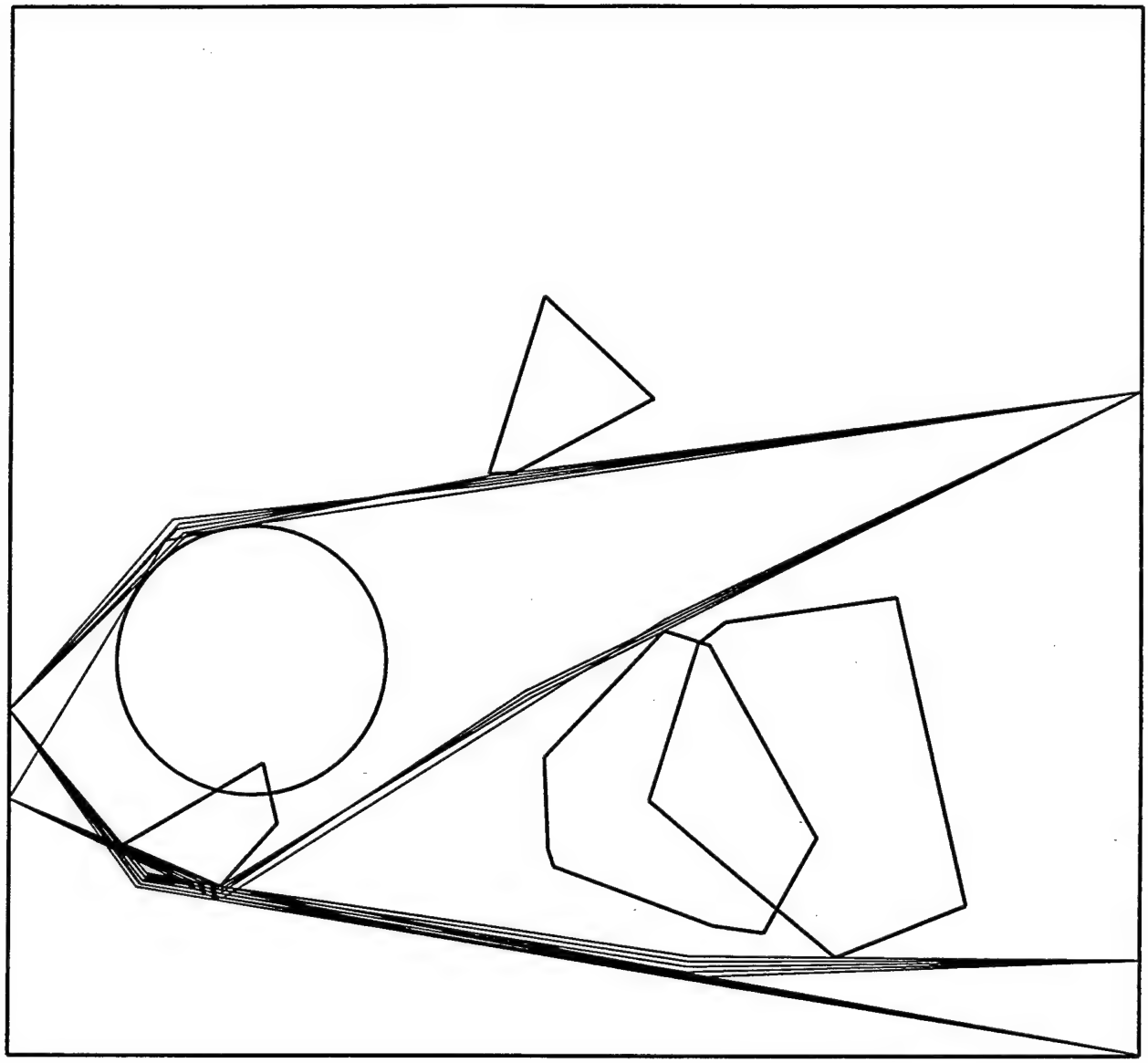


Figure 5: Display of Problem 2

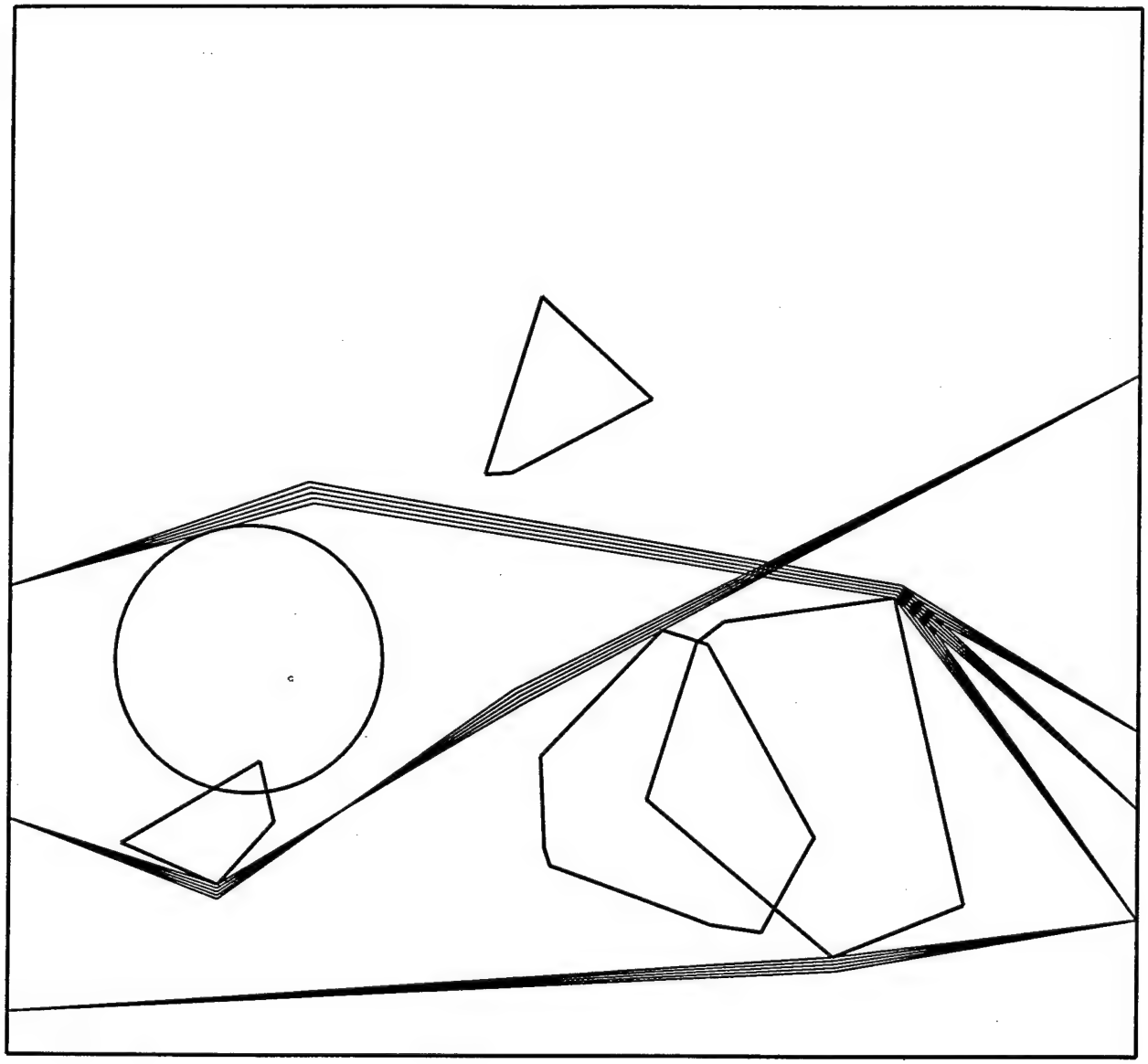


Figure 6: Display of Problem 3

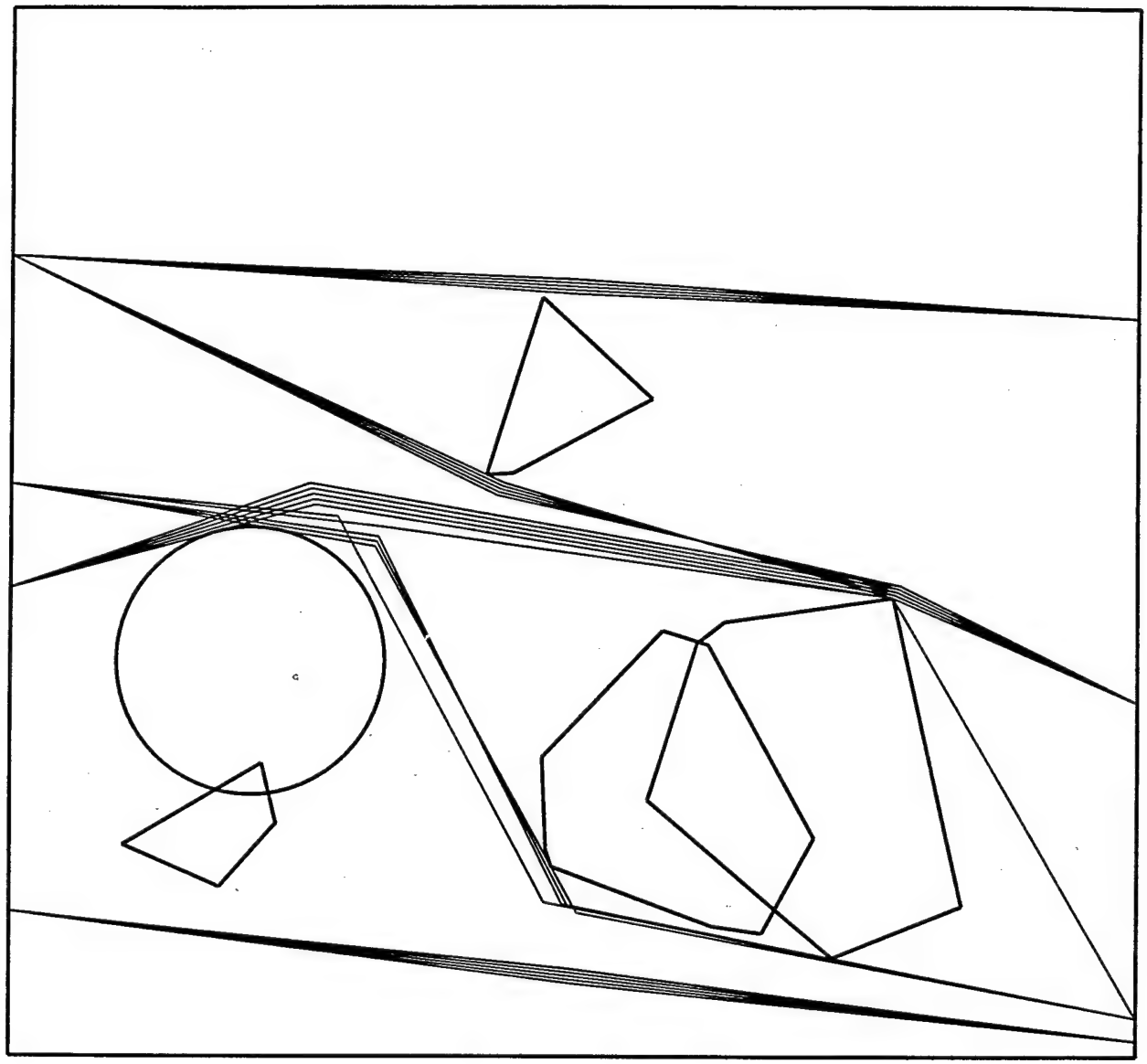


Figure 7: Display of Problem 4

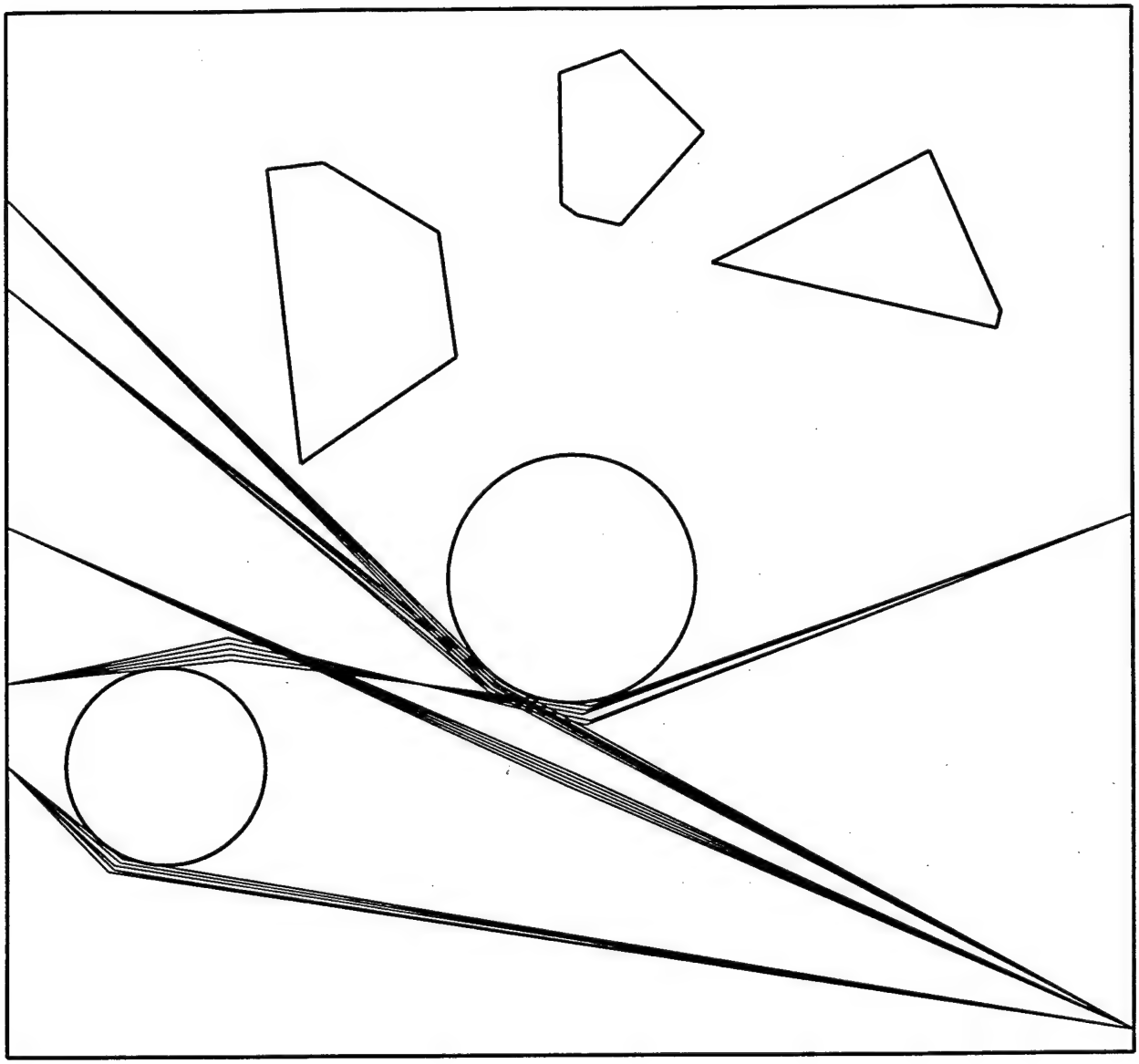


Figure 8: Display of Problem 5

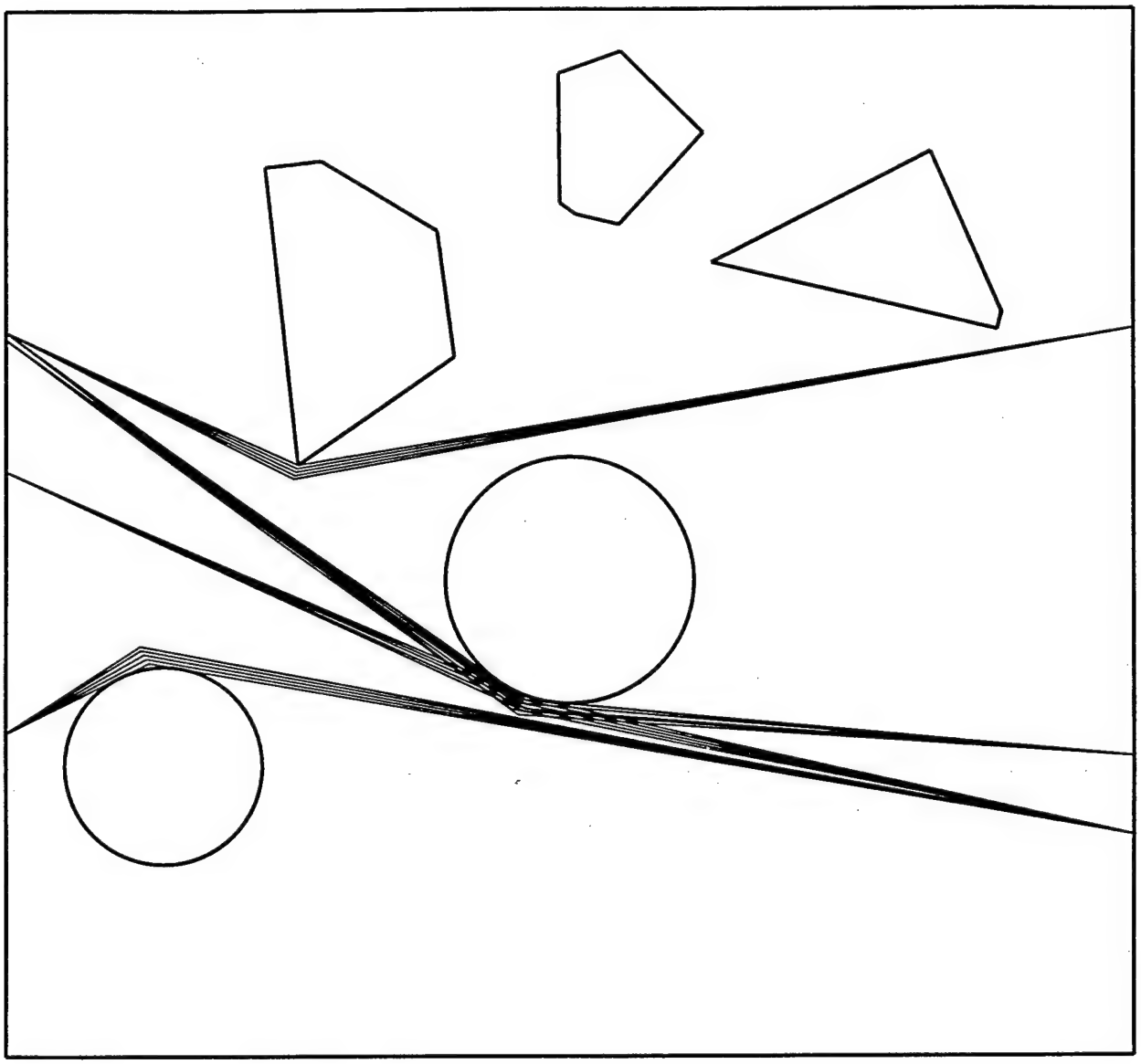


Figure 9: Display of Problem 6

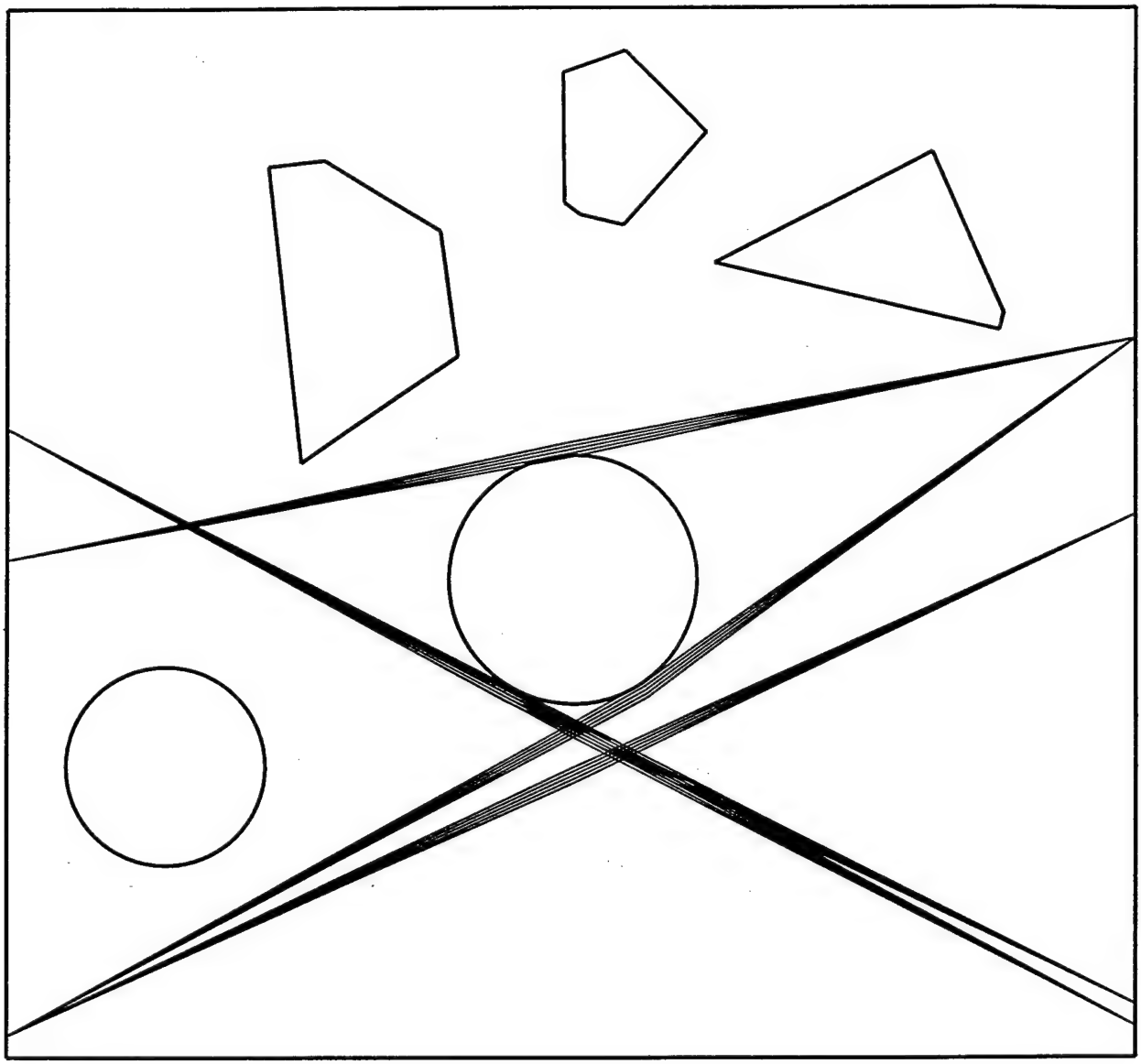


Figure 10: Display of Problem 7

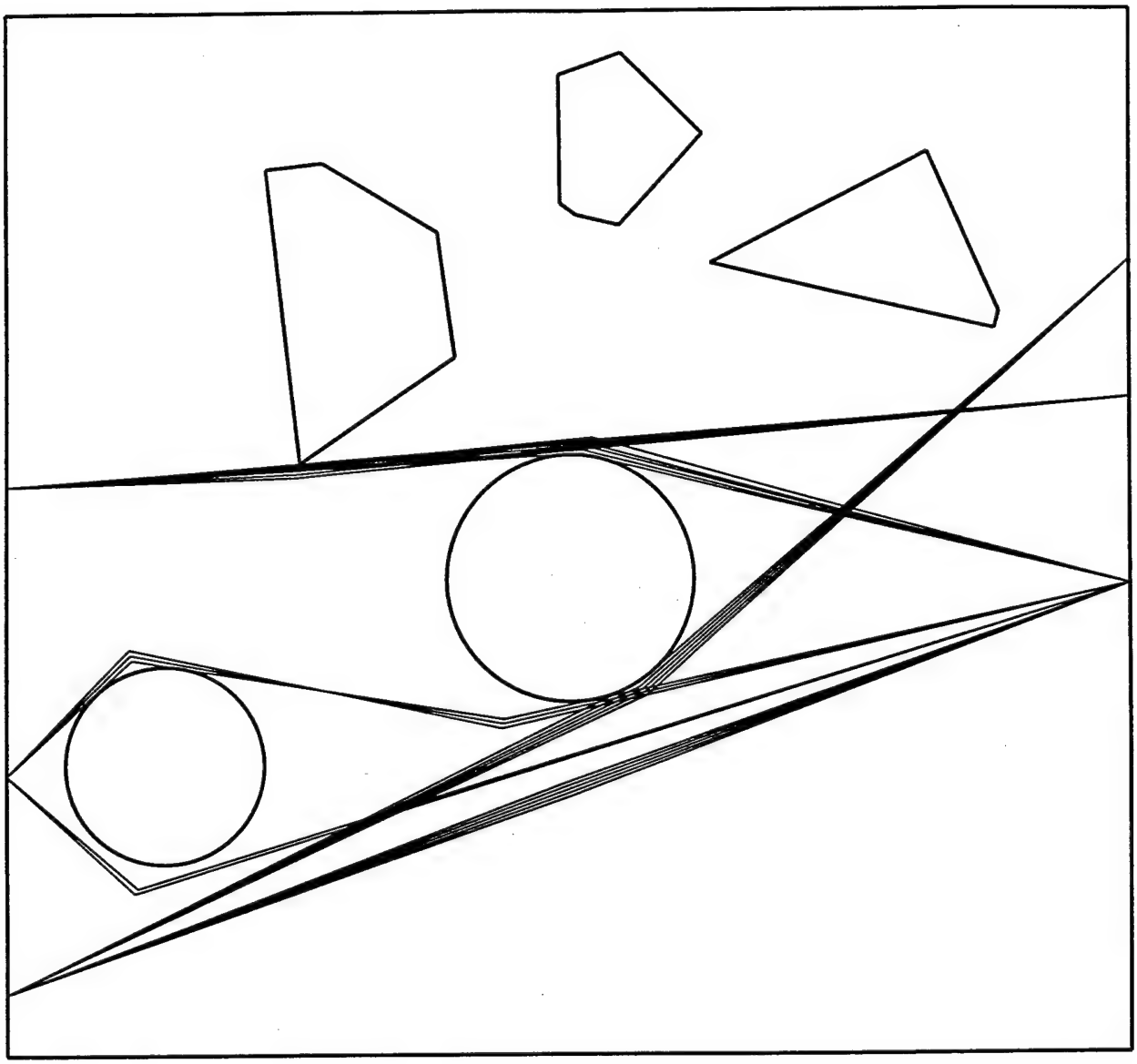


Figure 11: Display of Problem 8

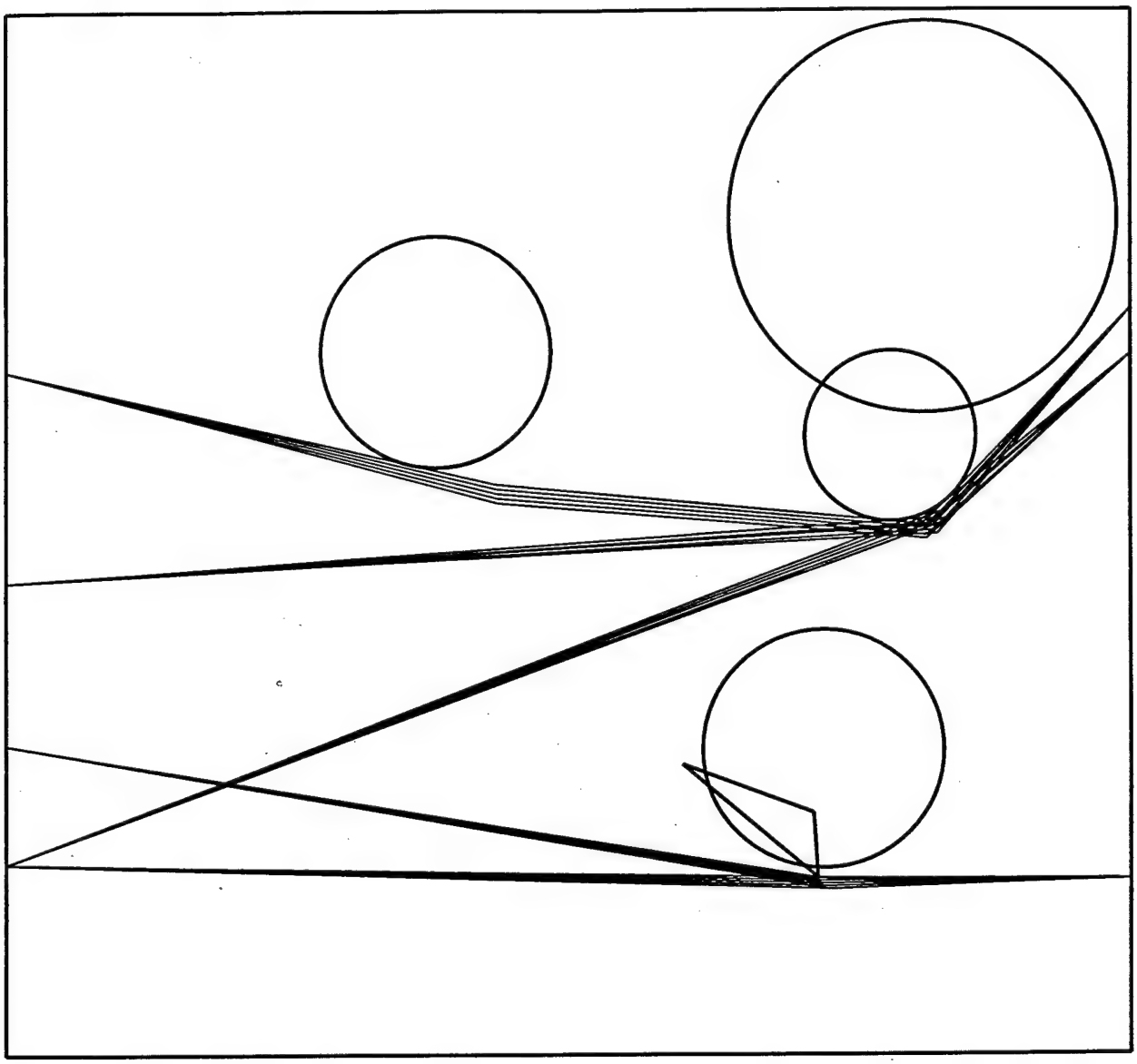


Figure 12: Display of Problem 9

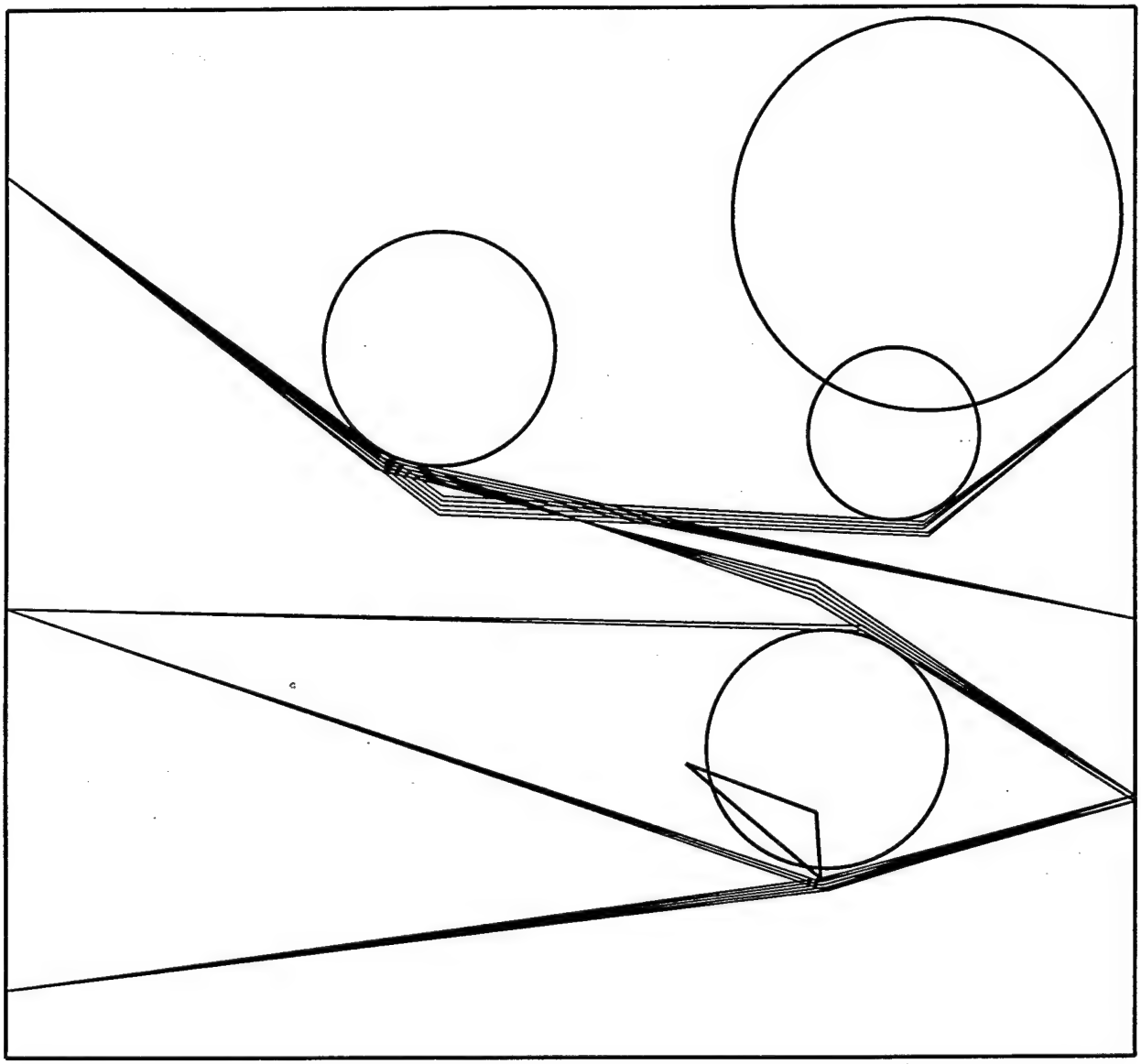


Figure 13: Display of Problem 10

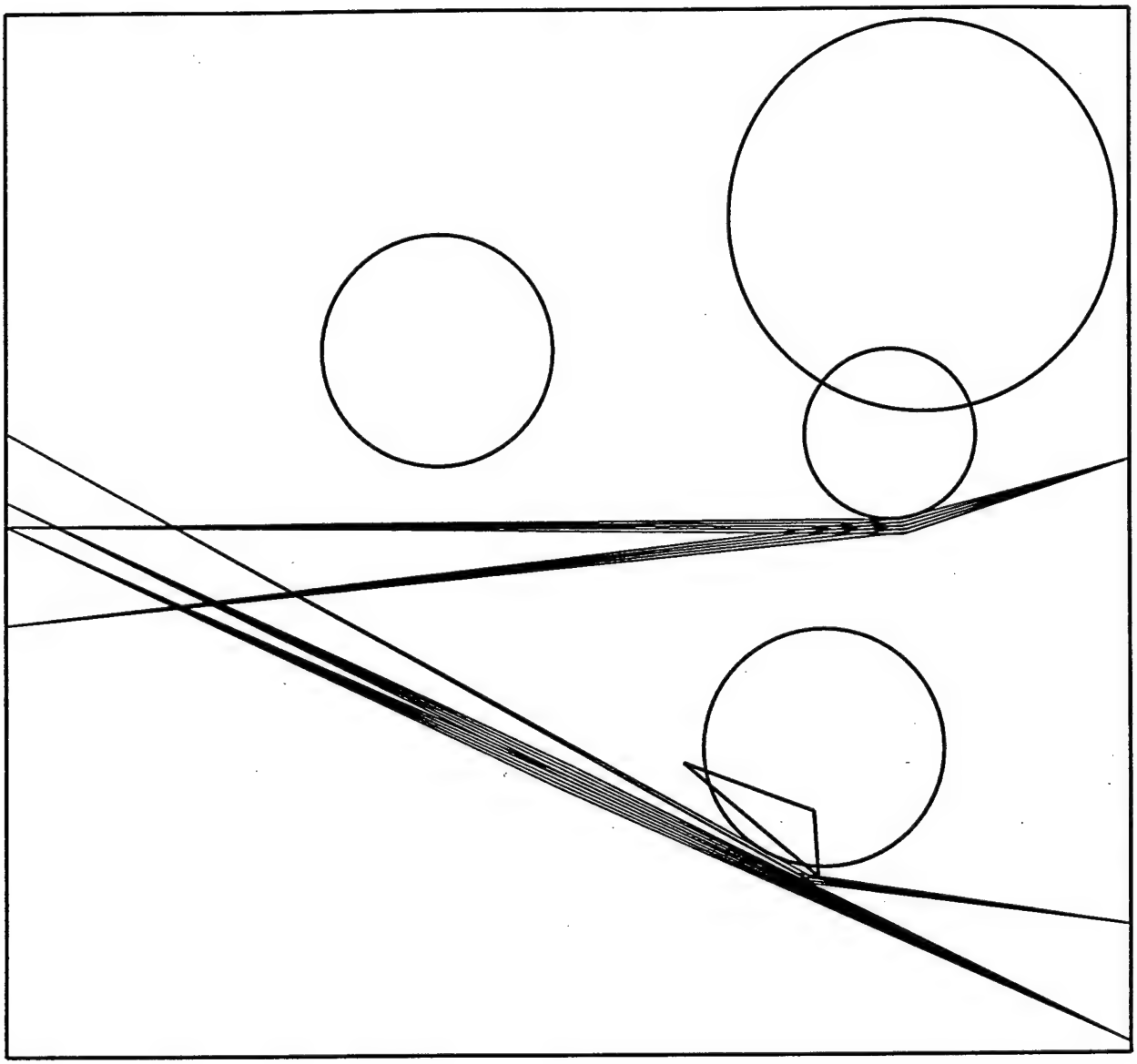


Figure 14: Display of Problem 11

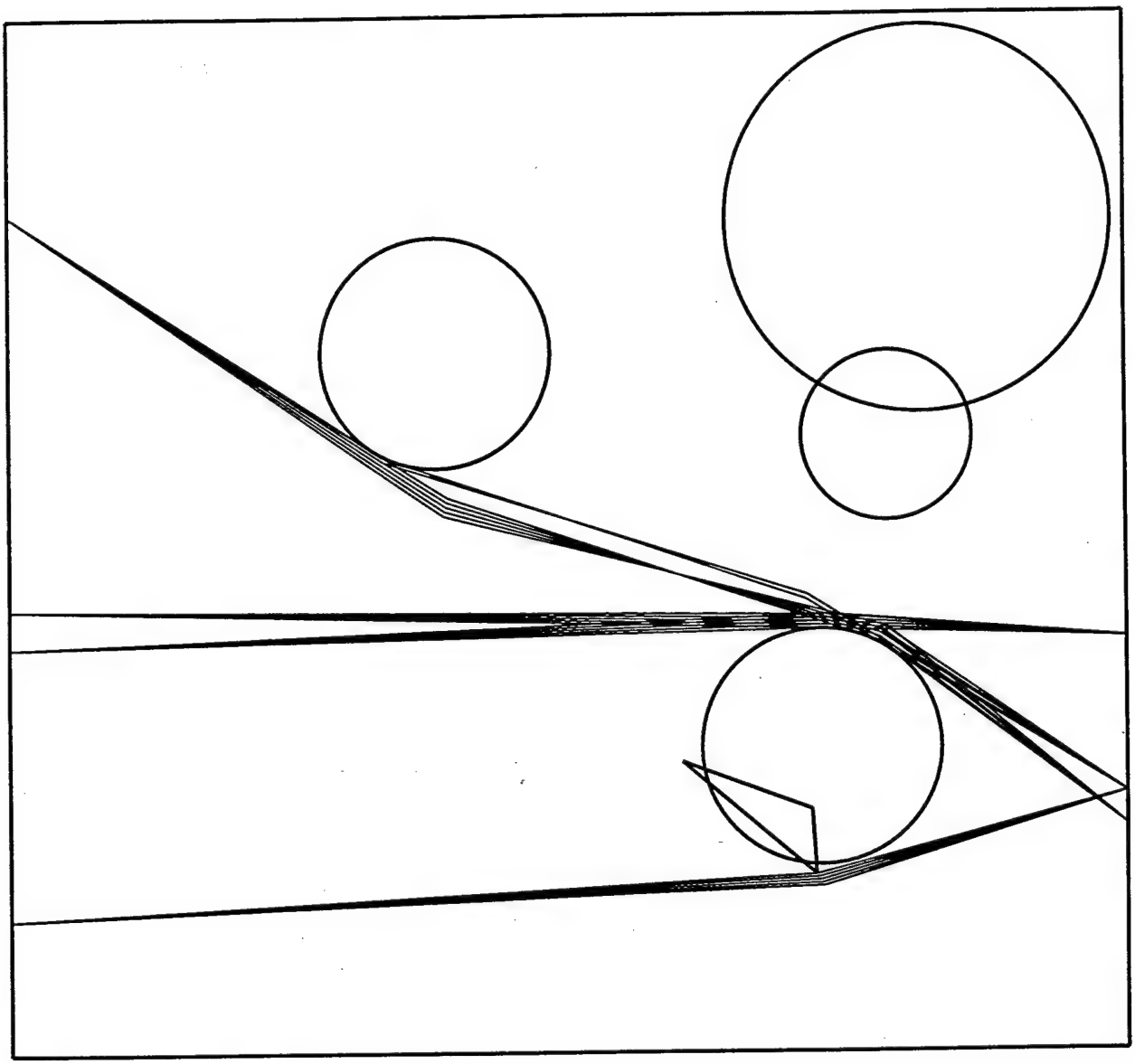


Figure 15: Display of Problem 12

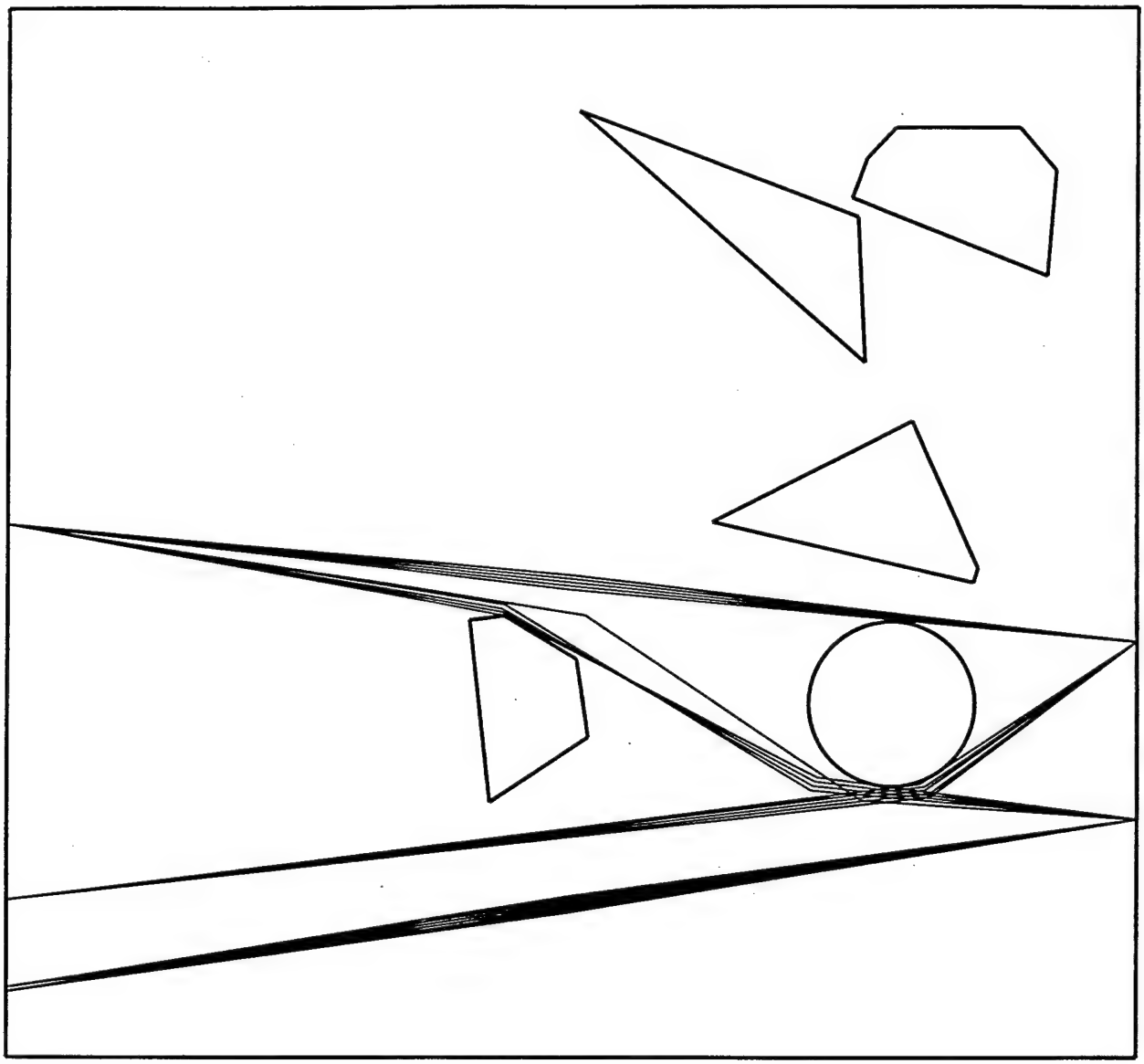


Figure 16: Display of Problem 13

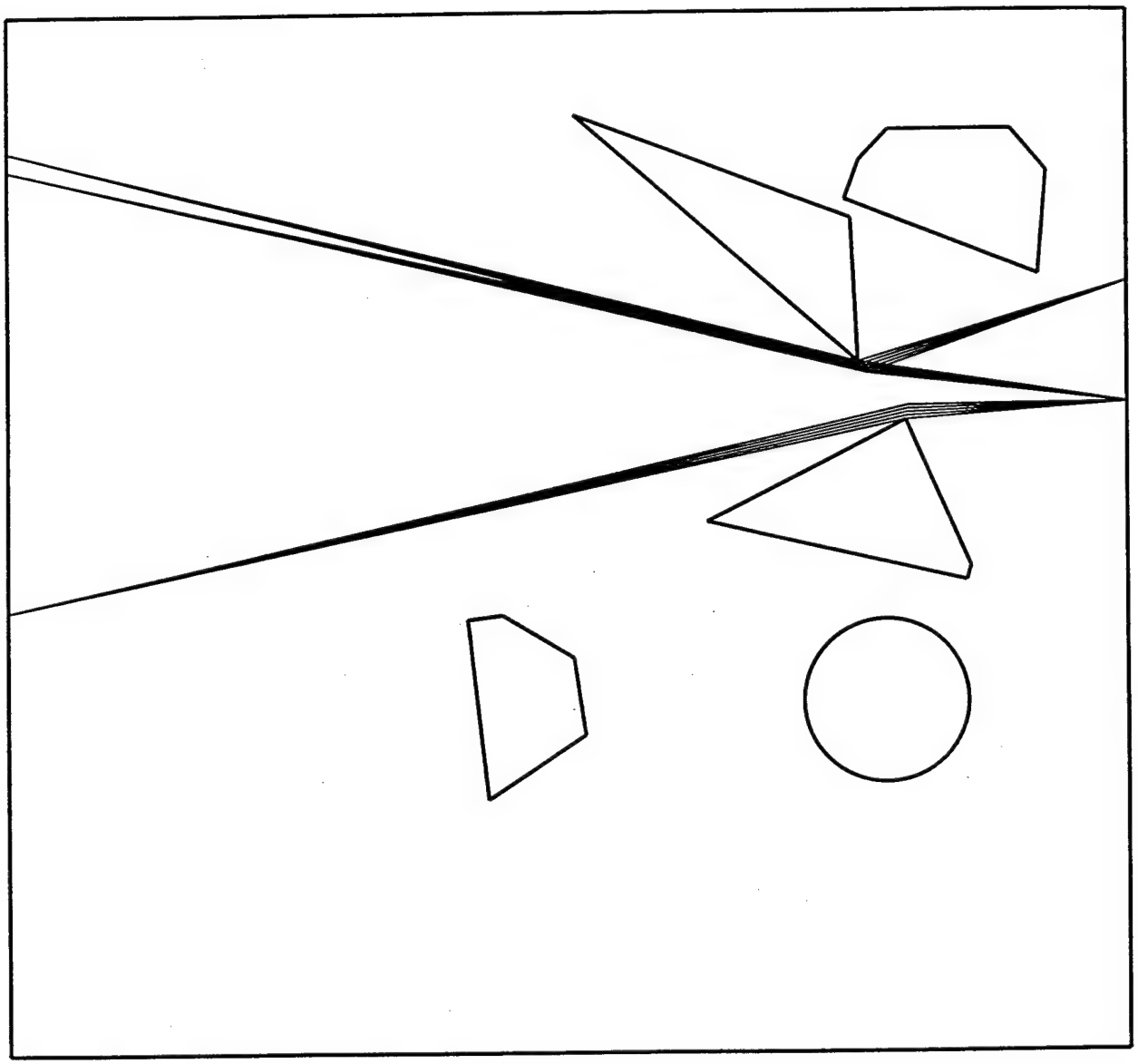


Figure 17: Display of Problem 14

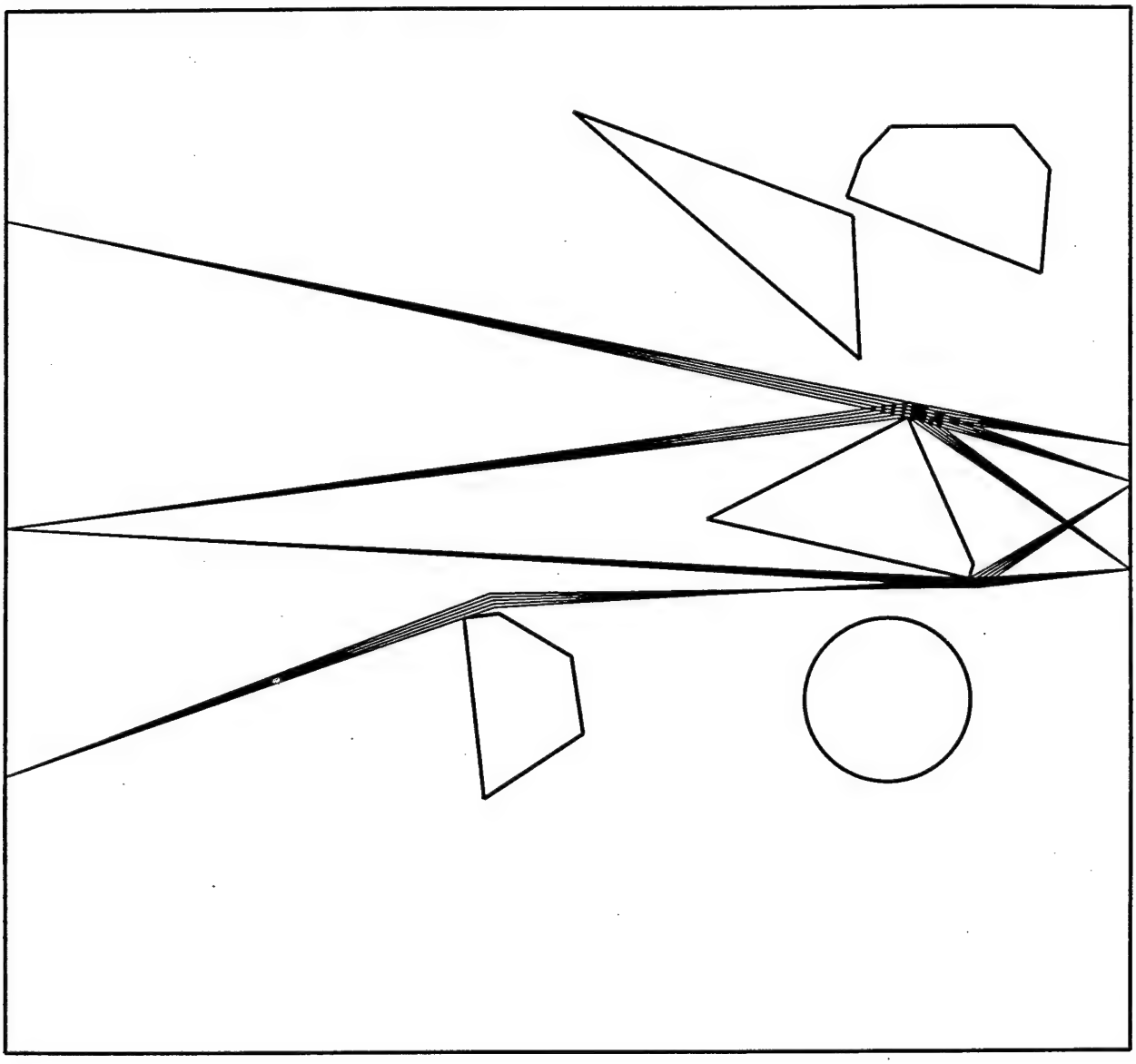


Figure 18: Display of Problem 15

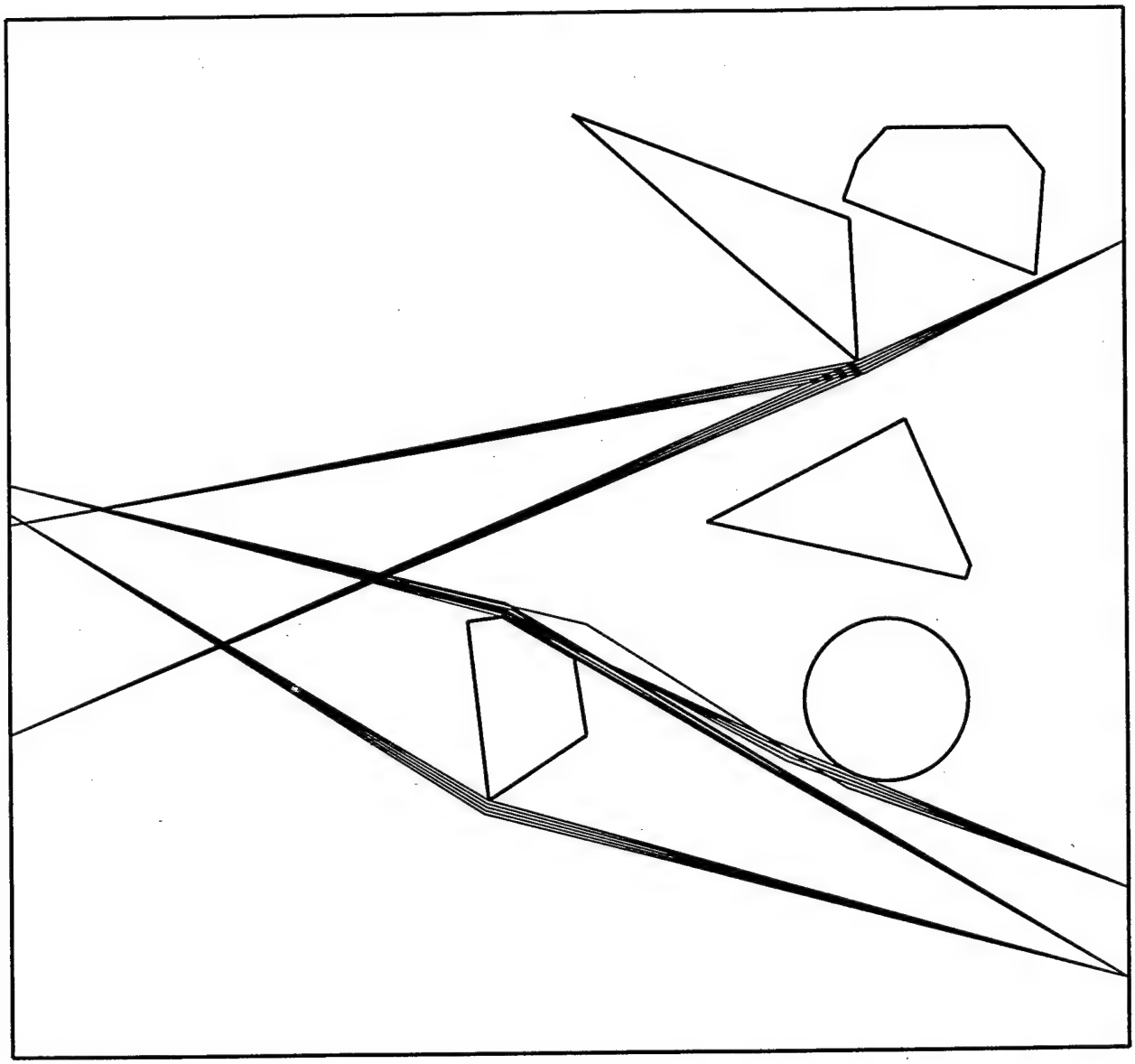


Figure 19: Display of Problem 16

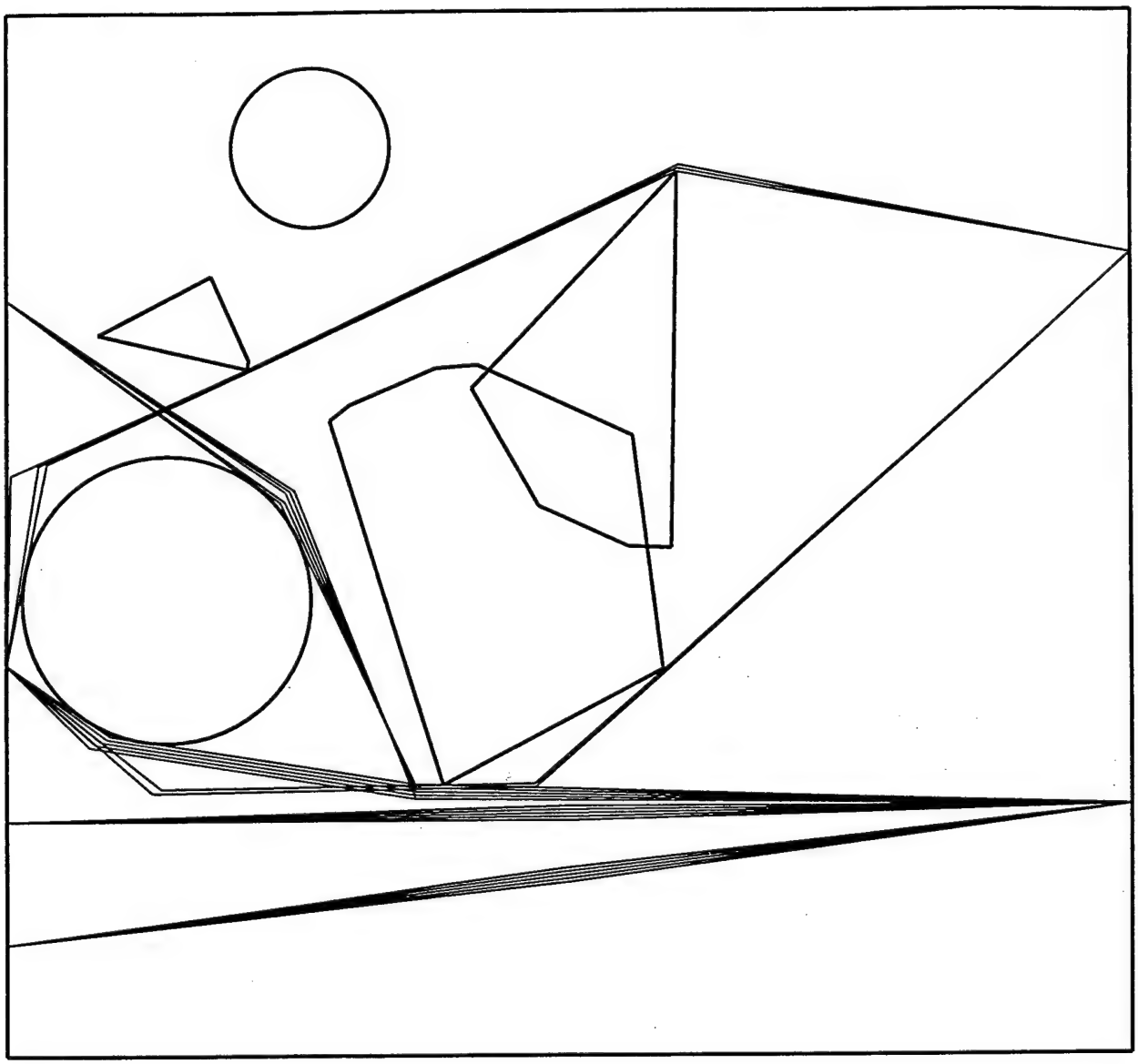


Figure 20: Display of Problem 17

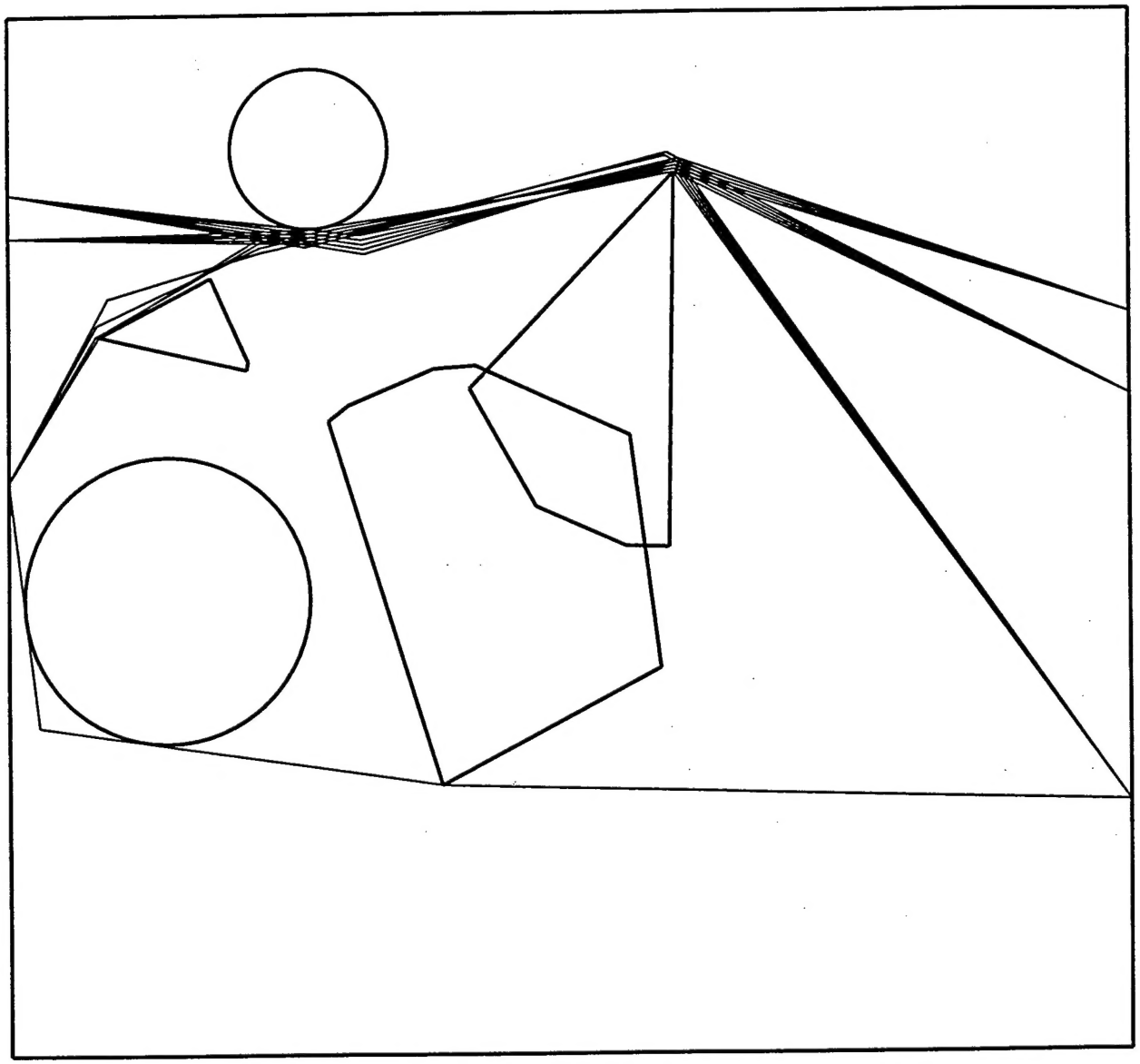


Figure 21: Display of Problem 18

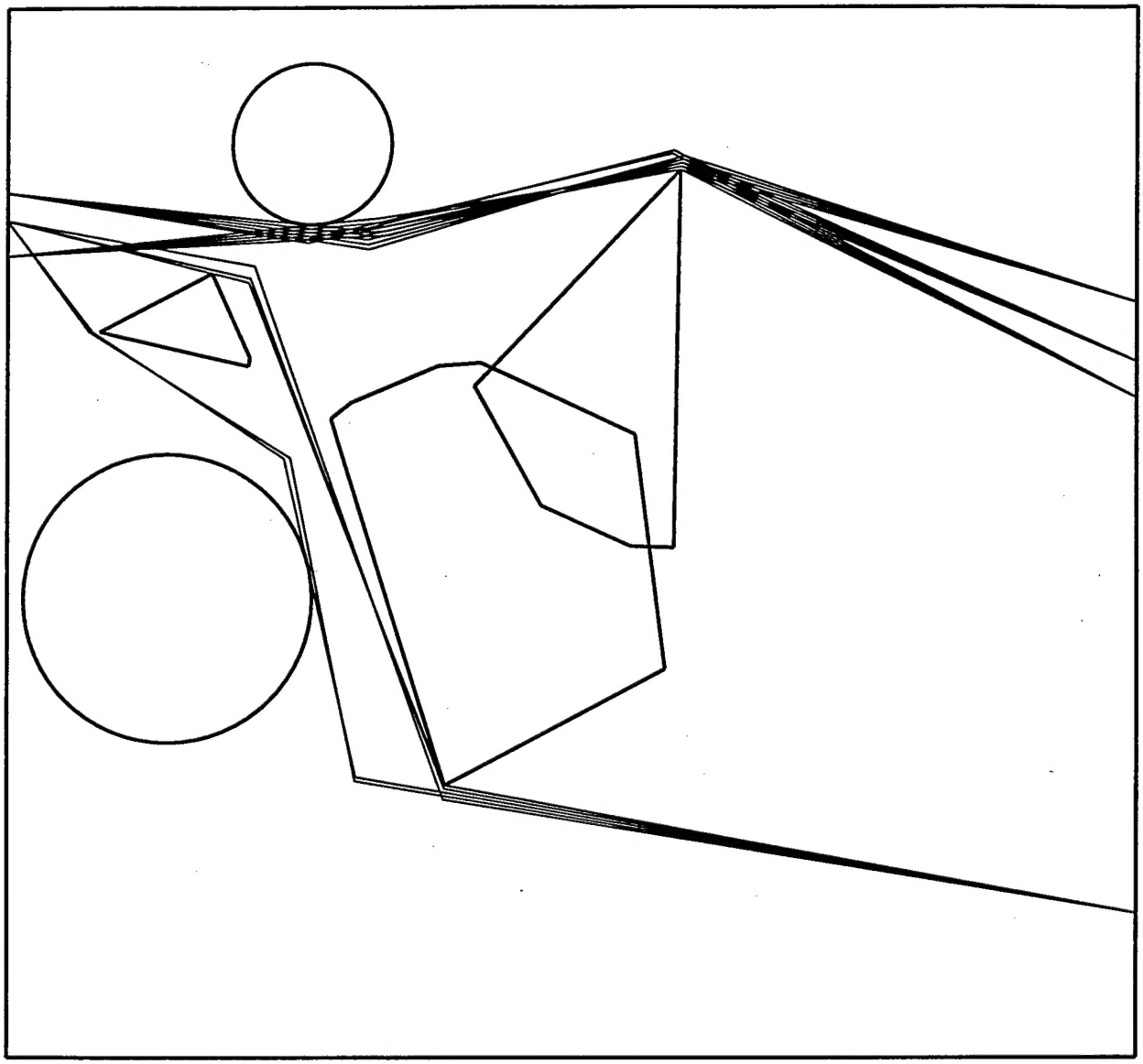


Figure 22: Display of Problem 19

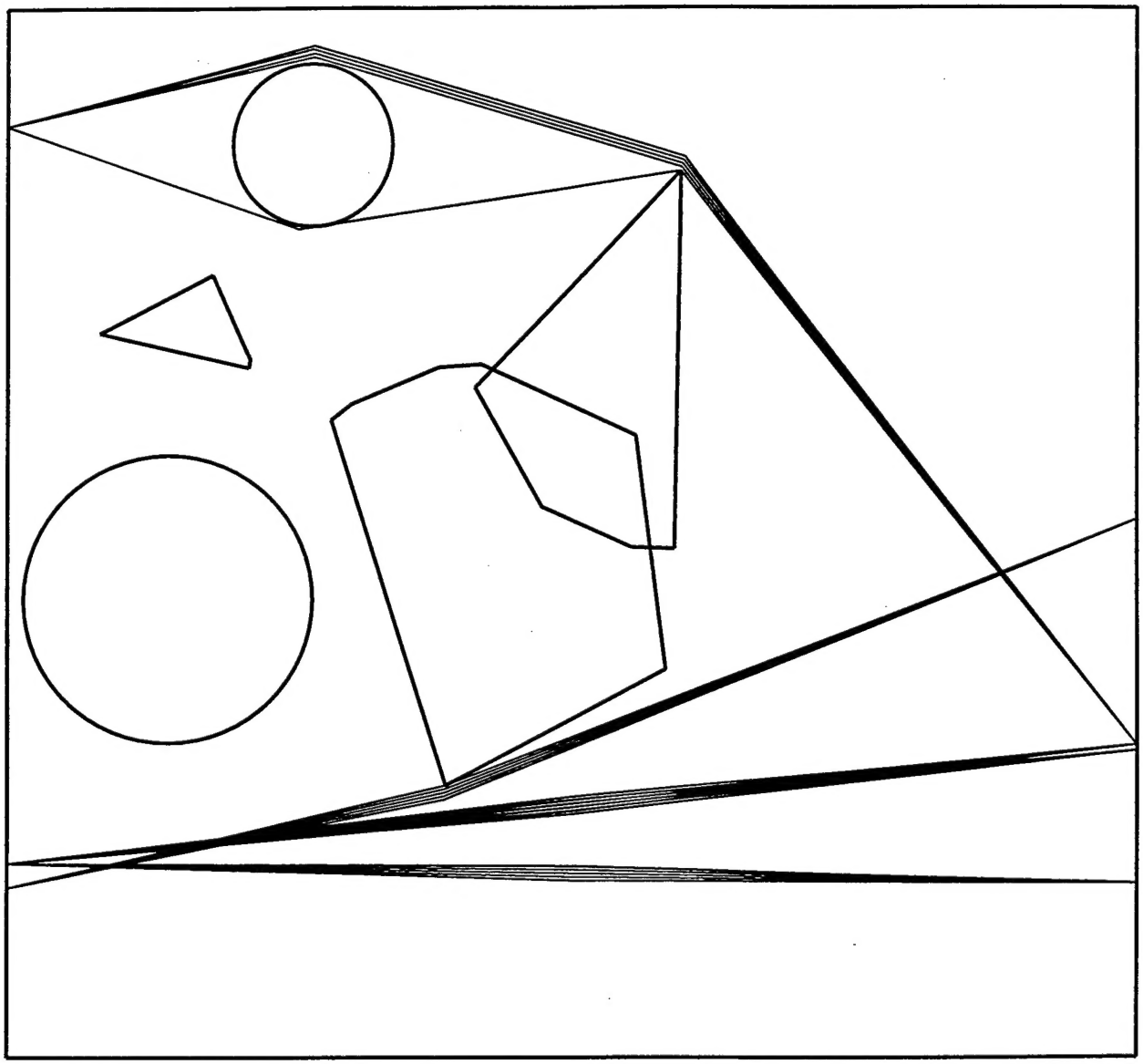


Figure 23: Display of Problem 20

Appendix C

Distribution List

Donald Wagner — 3 copies
ONR 311
Ballston Centre Tower One
800 North Quincy Street
Arlington, VA 22217 – 5660

Administrative Grants Officer — 1 copy
Office of Naval Research Regional Office
4520 Executive Drive Suite 300
San Diego, CA 92121 – 3019

Director, Naval Research Laboratory — 1 copy
Attn: Code 2627
4555 Overlook Drive
Washington, DC 2037 – 5326

Defense Technical Information Center — 1 copy
8725 John J. Kingman Road
SDTS 0944
Ft. Belvoir, VA 22060 – 6218

Carol Voltmer — 1 copy
Office of Research Administration
SMU
Dallas, TX 75275